A methodology for Control, Calibration and Characterization of quantum devices, applied to superconducting qubits

Shai Machnes,[1,*] Nicolas Wittler,[1] Federico Roy,[1] Kevin Pack,[1] Anurag Sasha-Roy,[2] and Frank K. Wilhelm[1]

[1] *Theoretical Physics, Saarland University, 66123 Saarbrücken, Germany*

[2] *Citizen scientist*

Efforts to scale-up quantum computation have reached a point where the principal limiting factor is not the number of qubits, but the entangling gate fidelity. Unfortunately, current control methodology offers no help in this regard: Models and control ansatz are simplified to allow analytic analysis, but in the process lose their capacity to accurately predict gate fidelities. The resulting low-fidelity pulses are then calibrated in-situ, resulting in higher-fidelity gates. Unfortunately, one typically does not learn anything about the system from calibration process, meaning there is no model which corresponds to the calibrated pulses. Alternatively, one may attempt to construct a model using detailed characterization procedures, but this is an ardours process, especially for large quantum processors.

However, to scale-up quantum processors one must reduce infidelities, which requires an understanding their causes, which implies an error budget, which, in turn, requires a detailed model of the system.

To rectify the situation, we are implementing a novel procedure of Combined Calibration and Characterization (C^3): An iterative combination of open-loop pulse optimization, model-free tune-up and iterative model fitting and refinement, utilizing a high-performance TensorFlow simulator. It will allow for a rapid, and largely automated bring-up process of QPUs. The result is a high-fidelity model, commensurate high-fidelity gates and a detailed error budget.

C^3 software is made available as an open-source project at q-optimize.org

I. THE PROBLEM

Scaling up quantum processors is a monumental task, requiring the field make progress on multiple fronts. Over the past few years, a race has emerged to produce the working QPU (Quantum Processing Unit) with the largest number of qubits. We argue that while great progress has been made on this front, the field is currently hampered not by qubit count, but by gate fidelities, which have not improved to a commensurate degree. As a result, it is insufficient gate fidelities which currently constrain what can be computed on quantum processors. Further, we argue this difficulty is the result of two related causes: First, that the methods to characterize and calibrate QPUs are cumbersome, and as a result, a full detailed characterization is rarely, if ever, performed, even for small QPUs. And second, that this leads to a lack of a detailed models of QPUs - models which would allow us to produce a detailed error budget, and thus pin-point the issues which must be resolved to improve fidelities. The goal of this work is to resolve these two issues, allowing fast and robust characterization and calibration of QPUs, facilitating the determination of a detailed error budget, enabling subsequent improvement of gate fidelities.

A case in point is Google's quantum supremacy [1] experiment - a 53 qubit quantum processor has performed a calculation in less time and requiring less power than would be required to perform the same computation using a classical (super-) computer. What is perhaps less well widely acknowledged is that the quantum error-free evolution constituted only 0.2% of the 3×10^7 shots (experiment runs) taken. And while such statistics are sufficient to distinguish between classical and partially coherent computation, this also proves that little is to be gained at this time from efforts to grow the number of qubits further, as current gate fidelities prohibit us from making effective use of larger devices or higher circuit depths. This is by no means unique to the above experiment. IBM has demonstrated gate infidelities on a similar scale [2], and cites a quantum volume [3] of 32, very loosely meaning that a QPU of 5 qubits would be able to perform a circuit of depth 5 before the total error rate exceeds one third. In fact one could argue that the fidelities demonstrated in 2014's [4] and 2019's [1] are comparable (although the latter are for simultaneous gates in a large 2D qubit array, while the former are in isolation).

One could argue that the relatively slow progress in improving gate fidelities can be traced back to an incomplete understanding of the causes of infidelity. And to understand the causes, a detailed model of the system is required. One could then ask "what if" questions using modified models (e.g. how would fidelities improve in control bandwidth is increased), which would, in turn, provide an error budget. Therefore, an intermediate goal is to arrive at a satisfactory model of the system. As "all models are wrong, but some are useful" [5], we shall define a Good Model as follows:

* shai.machnes@gmail.com

A *Good Model* is one which predicts the behavior of the system for the operations we wish to perform, to accuracies we care about.

For a QPU, the model is required to have predictive power for the range of feasible gate-generating pulses and for a sequence of several dozens of such gates, to a fidelity accuracy on the order of 10^{-5}. This model can then be used to produce an error budget, as shall be discussed later. To the authors' knowledge, no Good Model for a QPU has ever been published.

Without a Good Model, standard approached to improve gate fidelities, such as optimal control [6–8], cannot succeed, as inaccuracies of the model will inevitably lead to commensurate experimental infidelities of control pulses optimized for that model. Of course, this has been noted before, and methodologies to address this issue have been proposed, such as Ad-HOC [9], where model-based open-loop optimal control is followed by a model-free closed-loop experimental calibration of the control pulses (e.g. see CRAB [10]). Unfortunately, such an approached leaves one in a very unsatisfactory position: the system model is proven inaccurate by observing the pulses generated by the open-loop optimal control failed to deliver on the predicted fidelities, and while subsequently calibrated pulse shapes have a higher-fidelity, no commensurate model is at hand.

Conversely, if a Good Model is known, the task of generating optimal gates for the hardware using optimal control is comparatively trivial, and such control pulses should not require further closed-loop calibration. An error budget also follows. Therefore, extracting a Good Model efficiently and in a highly automated manner is the key to improving gate fidelities and the next step of QPU scale-up. The methodology and tools reported here, Control, Calibration and Characterization (C^3), is aimed to do just that.

We note that the question of how calibration and characterization methods scale with the number of qubits is most likely a red herring, as it has been observed that the performance of the QPU as a whole can be determined by the performance observed for sub-systems, and that no large-scale correlated noise or similar scaling issues have been observed [1]. One can therefore proceed under the assumption that a large-scale QPU can be characterized as a set of overlapping nearest-neighbour or next-nearest-neighbour localized models.

The paper is organized as follows: In section II we review the currently available solutions to the above problem. Next we present our solution to the problem, C^3, in Section III. We provide examples of its utility in Section IV, including a model hierarchy, sensitivity analysis and error budget. This is followed-up by a more in-depth look into C^3 in Section V. The practical aspects of the open-source TensorFlow[11]-based implementation are presented in Section VI. Finally we conclude with a discussion of the effort's current status and long-term directions.

The software created to achieve this is released under the Apache open-source license at q-optimize.org.

II. PAST ATTEMPTS AT ADDRESSING THE ISSUE

The standard approach at addressing the lack of a Good Model, as defined above, is to perform a long list of characterization experiments, each designed to measure a different parameter of the model: Measure parameters of the readout resonator using frequency sweeps; qubit frequency measurements and T_1 require Rabi experiments (and with some extra effort the higher levels can be extracted); Ramsey experiments provides T_2 data (under the Markovian assumption, which is known to be an over-simplification [12, 13]); gate and process tomography, of which there are many variations [14–20]; measuring the control line response functions [21–26], the noise spectra [13, 27, 28], continuous drifts in system parameters [29–32], and discontinuous jumps in parameters such as T_1 [33, 34]; state Preparation and Measurement Errors (SPAM) can be extracted from Randomized Benchmarking (RB; e.g. [35]) or dedicated procedures, such as [36]; qubit cross-talk can be measured by the method described in [37, 38], etc.

The problem is that each such experiment requires a non-negligible effort, in setup, execution and data analysis. As a result, only a small fraction of experiments go through a comprehensive spectrum of characterization tools. Moreover, data analysis is highly non-trivial, as in many cases the characterization protocol and associated data analysis assume a system model which is not a Good Model, and therefore the model failures may lead to erroneous analysis results (e.g. using a Ramsey experiment to measure a not-actually-Markovian T_2, or using a simple RB variant which does not account for the ever-present leakage). Performing such a suit of characterization protocols on a QPU with dozens of qubits will also take so long that the systems parameters may drift in the interim, rendering the effort an exercise in cat herding. And while this Sisyphean task can be accomplished with Herculean effort, [39], such resources are not available to all but the largest experimental groups, necessitating a Promethean solution.

Ideally, a good framework to solve the above challenge will (a) automate both system characterization, open-loop model-based pulse optimization and model-free closed-loop pulse calibration, and use all these to generate a Good Model. Note that a Good Modelfor QPUs is required to predict gate fidelities to an accuracy $\mathcal{O}\left(10^{-5}\right)$, which is extremely challenging. (b) It follows that such a model must extend beyond Hamiltonians of multi-level qudits and their associated Lindblad

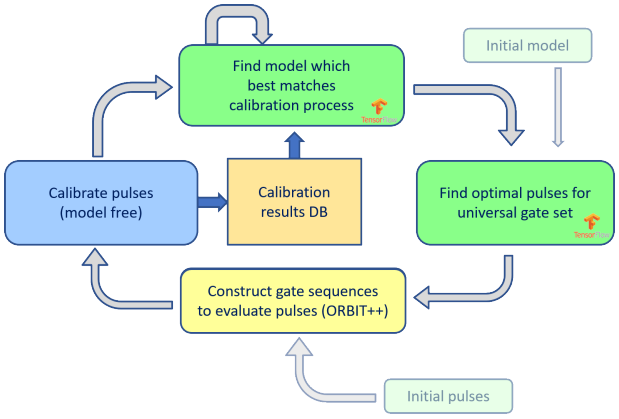


FIG. 1: C^3 is composed of 3 optimizations: C_1: Given a model and a task (e.g. gate generation), find optimal pulses to perform said task, C_2: Improve pulses in a closed-loop, model-free manner, C_3: Learn the model which fits all data gathered during C_2. If fit quality is poor, the model can be enhanced (e.g. add the possibility of noise on the control lines or spurious ZZ coupling between components), and the fit is repeated. C^3 performs all three optimizations, repeatedly, until they are all consistent in both model and pulse performance. The result is a Good Model and optimal fidelities for the given hardware. The process may begin with C_1 and a simplistic model, or with calibration data gathered in C_2. For more detail, see Sections V B, V C and V D, respectively.

terms, but also non-Markovian noise, control noise, State Preparation And Measurement (SPAM) errors, control line response functions and more. (c) Most often, the lack of model extends beyond uncertainty regarding model parameter values, but extends into which physical phenomena must be included in the model. Therefore, the framework must provide tools for exploration of possible additions and modifications to the model. Finally, (d) a good framework will merge naturally with existing experiments and lab procedures.

Not surprisingly, the community has been aware of the lack of such a framework. Several suggestions have been put forth to tackle the challenge, and they are reviewed below. We note that model determination has obvious overlaps with process tomography, which we shall not cover here, as tomography results in the description of the process (propagator), and not the model which creates it (Hamiltonian).

It is the authors' position that none of the proposals below meet the criteria for a good framework, described above. This work attempts to rectify the lacuna with the comprehensive approach taken by the C^3framework, presented in Section III.

III. THE PROPOSED APPROACH: C^3 - CONTROL, CALIBRATION AND CHARACTERIZATION

Once a Good Model is known, it is relatively easy to determine optimal pulses, error budget, etc. But finding the Good Model is exceedingly difficult, as it is a-priori unclear which physical effects the model must include (e.g. in transmons, we do not know if, or how many, rogue TLS-s are coupling to our qubits). Further, different aspects of the model may be accessible only when operating at the right frequencies, or require specific controls to manifest (e.g. long, spectrally narrow, Gaussian pulses are less sensitive to qudit inharmonicity). Therefore, in order to learn the model, we require data reflecting experiments conducted in the right regimes, so as to achieve good coverage of relevant system dynamics. And knowing which control fields will generate the desired dynamics requires a Good Model. The problem would seem to be circular.

The solution becomes apparent when we go back to the definition of a Good Model: "A Good Model is one which predicts the behavior of the system for the operations we wish to perform, to accuracies we care about". Luckily, we already have experimental protocols guaranteed to provide good coverage: single and multi-tone spectroscopy, gate calibration procedures and fidelity estimation protocols (e.g. Randomized Benchmarking). Therefore, C^3 will utilize these existing protocols (or slight variations thereof), and characterize the models using data collected during calibration and fidelity assessments.

As Control, Calibration and Characterization are inexorably linked, C^3 is comprised of three optimizations:

- C_1: Given a model, find the pulse shapes which maximize fidelity with a given target gate. This is the typical quantum optimal control task, and results are only as good as the model used. Pulse shapes may be constrained by an ansatz (e.g. Gaussian with DRAG [40] correction), or allow direct AWG parameterization.
- C_2: Closed-loop model-free calibration of the pulse shapes. Having this calibration model-free allows it to improve pulses beyond the limits of the model. All experiments performed and fidelities observed are stored for subsequent model extraction.
- C_3: Using all the measurement data collected during C_2, replicate all experiments and simulation, and optimize model parameters to minimize distance between simulation and experiment. If the convergence is poor, the model can easily be made more complex and the optimization repeated, until simulation and experiment match to the accuracy with which experimental data has been determined. This is the main role of the physicist.

One should view these three optimizations as stand-alone tools made available to the physicist exploring the system. And while the C^3 approach envisions using them in-sequence and as a loop, with the intended goal of finding a Good Model, this is by no means required. In addition, we provide tools to determine an error budget and the sensitivity of both the optimized model and control sequences.

How the three optimizations interact is determined by the overarching goal of minimizing wall time. Therefore, depending on the speed of simulation and data acquisition rate, it may be beneficial to run them all in parallel, or serially. With superconducting qubits, gate times $\mathcal{O}(20\,\mathrm{ns})$ and experimental repetition rate $\mathcal{O}(100\,\mu\mathrm{s})$, and simulations executed on a single (high core-count) PC, we have found it best to run the optimizations sequentially, so that closed-loop optimization and other data acquisition tasks never wait for open-loop simulation and optimization. Once C_3 concludes, we repeat C_1, etc. until the process converges and we have experimentally tested optimal pulses for the hardware, and a Good Model to explain them. We can then derive an error budget, identifying what limits current fidelities.

A major challenge of all three optimizations is the existence of local minima (see Fig. 9), and noise-induced false minima for C_2 and C_3. While choice of search algorithm alleviates the problems somewhat (see discussions in Sections V B,V C and V D), no magic bullet exists. To resolve the problem we adopt a method well known to experimental physicists: In complex systems, characterization and control must progress through a series of progressively refined models of the system and using progressively complex control pulses. In each step we build upon knowledge gathered in previous steps. From an optimization theory perspective, the simpler controls and models create a simpler, lower-dimension optimization landscape. And the carrying of knowledge from one step to the next ensures that all optimizations start fairly close to the optima, reducing the severity of the local minima problem. C^3 follows this methodology.

We note that in C^3 a model takes explicit values for its various parameters, and is *not* represented as a high-dimensional distribution over model parameter space. This choice is driven by classical computation-load considerations: a Good Modelof two qubits is likely to have dozens of parameters (see Section V A), with interdependencies which are more complex than can be reliably represented by simple error bars or a covariance matrix. Because the C^3 model is highly detailed, and, as consequence, associated simulations are non-trivial, we believe a full Bayesian approach to any of C^3 optimizations is not viable at this time.

The intertwining of control and characterization have also been raised in a more general control theory context. See, for example, [41–44].

C^3 software is released under the Apache 2.0 permissive open-source license at q-optimize.org.

IV. SYNTHETIC APPLICATION EXAMPLE

As an illustrative example we show the C^3 procedure in a synthetic context. We simulate an "experiment" using a "real" model that we temporarily forget and then recover using C^3. This demonstrated both the effectiveness of C^3 and how an actual experiment would implement it.
We start assuming a simple model and obtain optimal pulses using open-loop optimal control. Then, we proceed to calibrate those pulses on the "experiment"and use the calibration data to a learn a Good Model. We show that a simplified model cannot explain the behaviour of the "experiment" and systematically extend the model until it is rich enough to do so. Next, we perform again open-loop optimal control with the Good Model and demonstrate optimality by showing that repeating the calibration procedure doesn't lead to an improvement in experimental results.

The "real" model

We consider a single qubit, directly driven by an external field, modelled as a 3-level duffing oscillator. The Hamiltonian describing this system is

$$H/\hbar = \frac{\omega_q}{2} b^\dagger b - \frac{\delta}{2}\left(b^\dagger b - 1\right) b^\dagger b + c\left(t\right)\left(b + b^\dagger\right) \ , \quad (1)$$

where ω_q is the frequency, δ is the anharmonicity, b ($b^\dagger$) is the annihilation (creation) operator. To account for T_1 and T_2^* effects the dynamics are obtained by solving the Master Equation in Lindblad form. We include distortions in the line as a transfer function Φ, and calculate the effective control field $c\left(t\right) = \Phi\left(u\left(t\right)\right)$ from the input signal $u\left(t\right)$. Finally, the ideal populations p_i are distorted because of misassignment and calibration of the readout signal. We measure populations of $\tilde{p}_i = Ap_i + B$ and call A the *readout scale* and B the *readout offset*.
We assume that at the start of the C^3 procedure the parameters of the real model are only know to a certain degree of precision. For example, the frequency and anharmonicity are assumed to be 2 MHz and 1 MHz larger, respectively, than the real values.

Open-loop Optimal Control

We start with the simple modelof equation (1) but assume a perfect control line ($\Phi = 1$) and perfect readout ($\tilde{p}_i = p_i$). In theory, we can perform high fidelity gates by simply using DRAG [40] pulses

$$c\left(t\right) = \sin\left(\omega_q t\right)\Omega_{\mathrm{Gauss}}\left(t\right) - \cos\left(\omega_q t\right)\frac{\alpha}{\delta}\dot{\Omega}_{\mathrm{Gauss}}\left(t\right). \quad (2)$$

Here, Ω_{Gauss} is a Gaussian envelope, $\dot{\Omega}_{\text{Gauss}}(t)$ is its time derivative and the DRAG parameter α=0.5 is the value theoretically prescribed to eliminate leakage into the second excited state. Hence, we design a set of DRAG control pulses

$$\mathcal{G} = \left\{X_{\pi/2}, X_{-\pi/2}, Y_{\pi/2}, Y_{-\pi/2}\right\} \ , \tag{3}$$

to implement the unitary rotations $X/Y_\gamma = \exp\left(i\sigma_{x/y}\gamma\right)$.

Calibration

Using the analytically derived pulses as a starting point we begin the process of calibration on the "experiment". Hence, we use a gradient free optimizer and treat the simulated "real" model as a black box. The parameterization of the pulses, $\bar{\alpha} = (A, \alpha, \omega_{\text{off}})$, is given by the values of maximum amplitude A of Ω_{Gauss}, the DRAG parameter α and a frequency offset ω_{off} that is applied on the driving frequency ω_d.

As a figure of merit for the performance of the gates we will use the fidelity of ORBIT [45] sequences, i.e. single length randomized benchmarking (RB) [46] sequences $s_k(\bar{\alpha}) = \left(\prod_j^{m-1} C_{k,j}\right) C_{\text{inv}}$ composed of m Clifford gates. Here $C_{k,j}$ are the random gates sampled from the Clifford group C (for a single qubit $\|C\| = 24$), and C_{inv} is chosen so that $s_k = \mathcal{I}$ in the ideal case. Hence, the fidelity of one sequence is defined as the population in the ground state $|0\rangle$ after performing the sequence on a state initialized to $|0\rangle$.
We use the atomic operations from eq. 3 to construct the set of Clifford gates and from them construct $N = 20$ RB sequences of length $m = 35$. To optimize the goal function

$$g(\bar{\alpha}) = \sum_k (g_k)/N \ , \text{with } g_k = 1 - |\langle 0| s_k(\bar{\alpha}) |0\rangle|^2 \ . \tag{4}$$

we use CMA-ES [47], a gradient-free search that uses batches of guesses to converge to a minima, as its done in [48]. The results of this calibration are shown in Fig. 2. Each blue dot in the plot is equivalent to an evaluation of the goal function g for a different set of $\bar{\alpha}$ parameters. At each evaluation the algorithm trials 20 sets of such parameters before sampling 20 new sets based on the obtained results. After convergence of the search the set $\bar{\alpha}$ with the best value of g is chosen to be optimal. To better understand the improvement we perform a full RB experiment for the initial and final values of $\bar{\alpha}$ and obtain an error per Clifford of ... and ... respectively.
It becomes here evident that, given an inaccurate model, the pulses deemed optimal by open-loop optimal control are not, in fact, performing and can be significantly improved on. Furthermore, as the starting guess of $\bar{\alpha}$ is far from the minima, a large number of evaluations is required during calibration to converge to high fidelities.
In the same figure in green, is shown the equivalent calibration process when the initial guess for $\bar{\alpha}$ is obtained by open-loop optimal control using the model that we learn in C_3. Noticably, the pulses' fidelity is, to begin with, alerady better than the best result from the previous calibration. Furthermore, the pulses are already optimal as the calibration procedure doesn't improve on the initial guess' fidelity.
The resulting populations from each of the attempted sequences $s_k(\bar{\alpha})$ is recorded and used to learn a Good Modelfor the experiment. We are in the position where we have calibrated pulses that work but lack a model that predicts can tell us why.

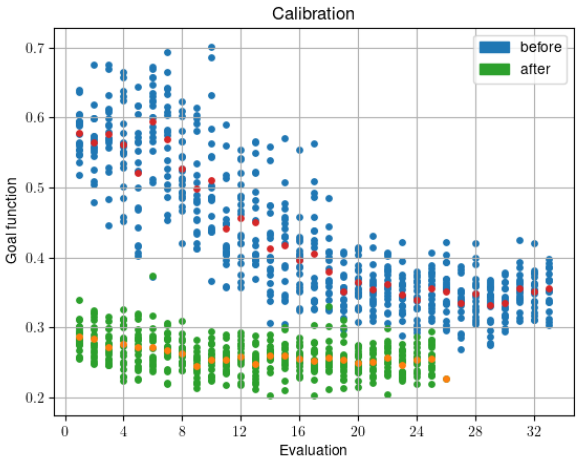


FIG. 2: Black box calibration process on the "experiment", simulated by the "real" model. The blue (green) dots represent the values of the ORBIT goal function before (after) learning, and the red (orange) dots show their average. After the model is correctly learnt the goal function of the pulses suggested by C_1 are both: better than the previous best value; already at an optimal position.

C_3 Model matching

We, now, demonstrate the model learning step of the C^3: we use the information obtained during calibration to improve our model of the system. From C_2 we stored a record of pulse parameters $\bar{\alpha}_j$ and corresponding goal function values $g_j = g(\bar{\alpha}_j)$. We should be able to match, up to measurement noise, these values in simulation, assuming the model is accurate enough. Formally we select n_j data points from the record and minimize

$$f(\bar{\beta}) = \sqrt{\sum_j^{n_j} \left|m_j(\bar{\beta}) - g_j\right|^2 / n_j} \ . \tag{5}$$

where the set of $\left\{m_j(\bar{\beta}) = m(\bar{\beta}, \bar{\alpha}_j)\right\}$ are the results given by simulating the record of pulses for a given assumption of model parameters $\bar{\beta}$ and g_j are the the

calibration values. We begin with our simple model that we assumed in C_1 (closed system dynamics from Schrødinger equation, no distortion of the line and perfect measurement) and try to find the frequency and anharmonicity of the qubit (here $\bar{\beta} = \{\omega_q, \delta\}$). To do this, we perform a gradient search on $f(\bar{\beta})$, with gradients obtained via automatic differentiation. In figure 3 are shown the values for ω_q and δ for this simple model as a function of evaluations of f, instead, as red dashed lines are the actual values from the "real" model. It is apparent that the values weren't well matched. Indeed, in figure 5 we see the convergence of the RMS as a function of iterations of the algorithm shown in blue for this simple model and notice that it wasn't much improved. Furthermore, from figure 4 we can see how there is no correlation between m_js and g_js.

This is a crucial point in the C^3procedure: it is the role of the physicists to determine what elements in the model of the system are missing to correctly capture the dynamics of the experimental results. In this case, we first assume that some of the mismatch is due to lack of open system dynamics in our simulation, hence we introduce this complexity to our model: we now calculate dynamics by solving the master equation in Lindblad form to account for T_1 relaxation/thermalization effects, and T_2^* dephasing effects. Once again the initial guesses for values for T_1 and T_2^* are similar but different from what we know them to be in the blackbox "real" model. When we attempt learning the parameters $\bar{\beta} = \{\omega_q, \delta, T_1, T_2^*\}$ we find that the final match is still not satisfactory, as denoted by the green line in Fig. 5.

In the next attemp we increase model complexity by adding two important features: a transfer function of the line, and measurement calibration effects. We now arrived at a model of the same structure as the "real" model we put in the black box, but its parameters are onec again either unknown or rough guesses. The C_3 procedure on this model converges to a low RMS value

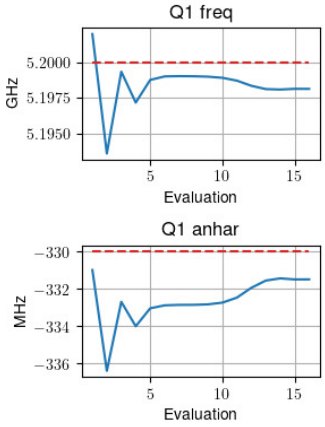


FIG. 3: Trying to learn parameters on a over simplified model.

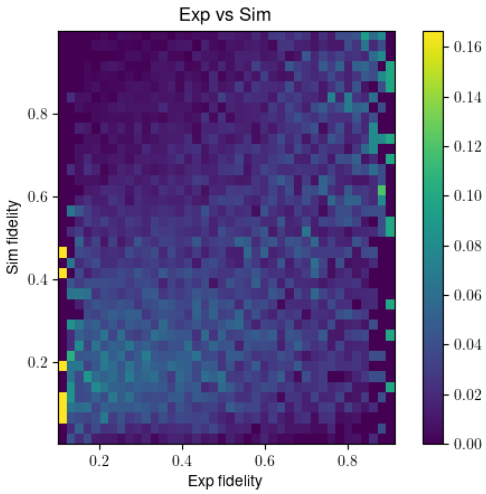


FIG. 4: Correlation between real g_j and simulated m_j after optimizing the parameters $\bar{\beta}$ for the simple model

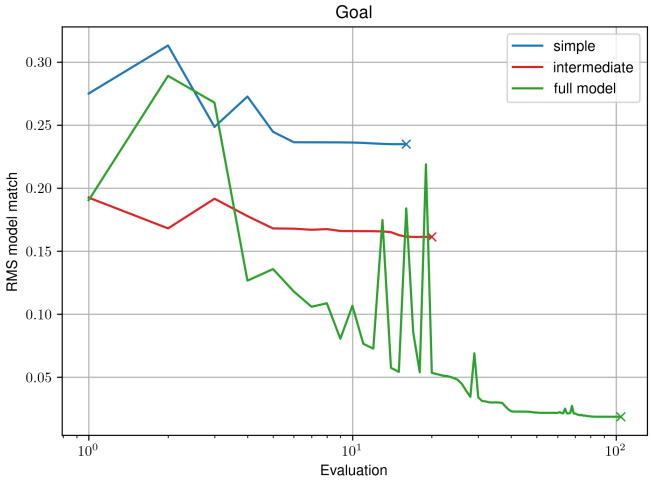


FIG. 5: Learning progress on a hierarchy of models: simple model (blue), intermediate model (red) and full model (green). The crosses show the points when the stopping condition of L-BFGS is met. The goal function (y-axis) is the RMS between real g_j and simulated m_j values.

(orange line in Fig. 5) and shows correlation between model predicted fidelities and the record (Fig. 7). We also verify that most model parameters have approached their "real" value, as indicated in Fig. 6. Noting that the "real" rise time parameter is not recovered while still matching to high accuracy leads to the conclusion, that this parameter has little impact on the performance of our control pulses. This observation is directly bound to the applied control pulses and does not exclude the possibility that other pulses might reveal some dependence on this parameter. The extracted information about the model can only be as sophisticated as the pulses that were used to interrogate it. In this case the pulses are

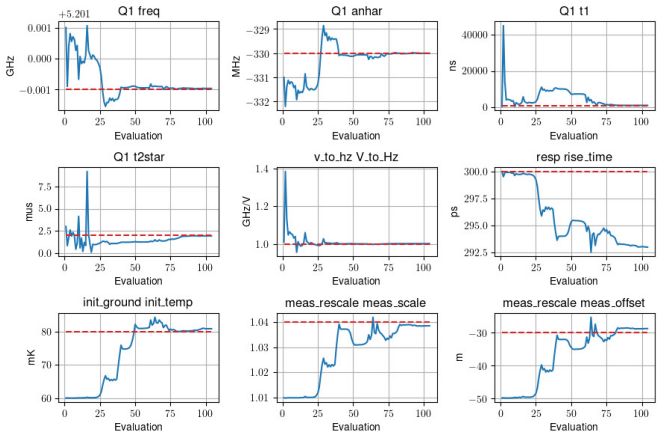


FIG. 6: Convergence of model parameters.

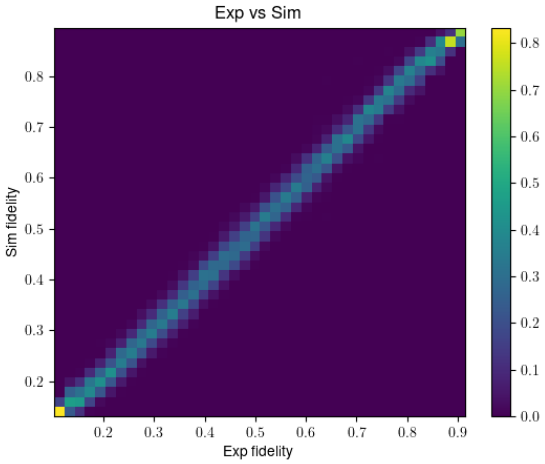


FIG. 7: Correlation between parameters.

symmetric so any power lost during the risetime on one side of the pulse is recovered on the other side (this can be seen in Fig. 8a), and thus only effects the resulting dynamics to second order. Highly asymettrical pulses should make the effect of the rise time more apparent.

Repeat Open-loop and Closed-loop Optimal control

Now that we are confident that we have learnt a good model, we can perform open-loop optimal control again and obtain pulses that work on the "experiment". We finally calibrate the good pulses to see that we are already at the minima of ORBIT fidelity.

Analysis: Robustness and error budget

Having arrived at a good match to the model allows for sophisticated analysis. As noted previously, the goal function seems less sensitive to certain parameters. More systematically, we can estimate this sensitivity by performing one-dimensional scans of the model parameters and observing what deviation from the optimum is needed to degrade performance by a fixed value. We then assign the corresponding error bar to the parameter.

Similarly, we can identify the limiting factors in experiment by exploring "what-if" scenarios.

- Higher AWG resolution
- longer T_1
- larger anharmonicity

V. C^3 IN DETAIL

A. Simulator core

Implementing the C^3 methodology in software requires at its core a powerful physical simulator. In the open-loop optimal control stage (C_1) as well as during model matching (C_3) it is desirable to use gradient-based optimization algorithms, the simulator therefore needs to provide the physical state of the system after applying some control signals. Since it is not practical to provide analytical gradients for every component of every conceivable model, we chose to use a numerics package that allows for automatic differentiation. With C^3, simulation of the system dynamics is performed using TensorFlow which allows for automatic differentiation: the simulation is formulated as a network, with well defined inputs (e.g. control and model parameters) and outputs (goal function values), connected by a large number of nodes, each performing a relatively simple operation (e.g. matrix exponentiation). TensorFlow is then capable of using the chain-rule, writ-large, and computing the Jacobian of the calculation - the gradient of each of the network outputs with respect to the network inputs (the origins of this capability is the back-propagation learning process of neural networks).

The disconnect in previous application of optimal control lies in the fact that experimentally accessible quantities are usually not the quantities that appear in a microscopic Hamiltonian model. For instance, several layers of translation lie between generating a pulse using arbitrary waveform generators (AWG), transmitting it through electrical wiring across different stages of cooling until it creates an electrical field that couples to the dipole moment of a superconducting circuit.

Formalizing this translation, shown in Fig. 8b is a fundamental task of the Good Model, since it connects experimental observation with theoretical understanding.

The simulation allows for the specification of control signals $\varepsilon_k(t)$ as analytical functions, Fourier decomposition or direct, piecewise-constant parameterization. To create experimentally feasible control fields $u_k(t)$ we take into account the sampling rate of the waveform generator producing the envelope signal $\varepsilon_k(t) \to \varepsilon_{k,i} = \varepsilon_k(t_i)$, where the $\{t_i\}$ are the AWG samples. These samples $\varepsilon_{k,i}$ represent voltages being applied to the control line and will thus exhibit a *rise time* τ that the electronics need to change from the value of the previous sample to the current one. We model this by applying a convolution of with a normalized Gaussian

$$\begin{aligned} \tilde{\varepsilon}_k(t) &= \int_{t_0}^{t_f} \mathrm{interp}\left(\{\varepsilon_{k,i}\}\right)(t)\, G(t_f - t)\, \mathrm{d}t \quad \text{with} \\ G(t) &= \frac{1}{N} \exp\left(-\frac{(t-\tau/2)^2}{8\tau^2}\right) , \end{aligned} \tag{6}$$

replicating a charging capacitance, while interpolating the pixelated signal

$$\mathrm{interp}\left(\{\varepsilon_{k,i}\}\right)(t) = \{\varepsilon_{k,i} \,|\, t_i \le t < t_{i+1}\} \tag{7}$$

to higher resolution for simulation. An I/Q-Mixer combines this envelope with a local oscillator signal of frequency ω_{lo} to

$$u_k(t) = I_k(t) + Q_k(t) \tag{8}$$

where the in-phase and quadrature components are assigned

$$\begin{aligned} I_k(t) &= \tilde{\varepsilon}_k(t) \cos(\omega_{\mathrm{lo}} t) \cos(\alpha_{xy}) \\ Q_k(t) &= \tilde{\varepsilon}_k(t) \sin(\omega_{\mathrm{lo}} t) \sin(\alpha_{xy}) \ , \end{aligned} \tag{9}$$

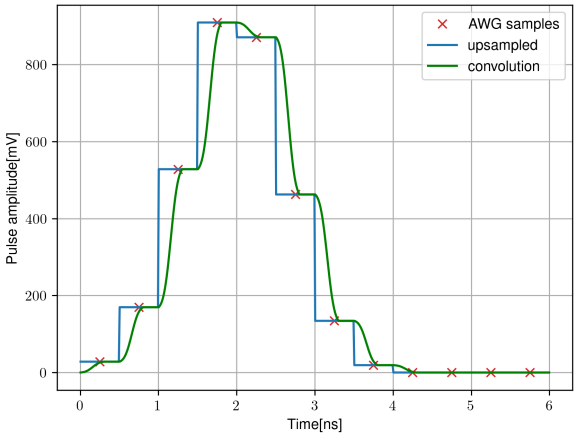
by a control parameter α_{xy}. The whole process is shown with an example in Fig. 8a. As noted in [49], in practice there will be additional errors during the mixing that could be considered here.

In transmitting this signal to the experiment various distortions can occur, modeled by a response function Φ that also converts from a line voltage to a field amplitude $c_k(t) = \Phi(u_k(t))$ that appears in the Hamiltonian model

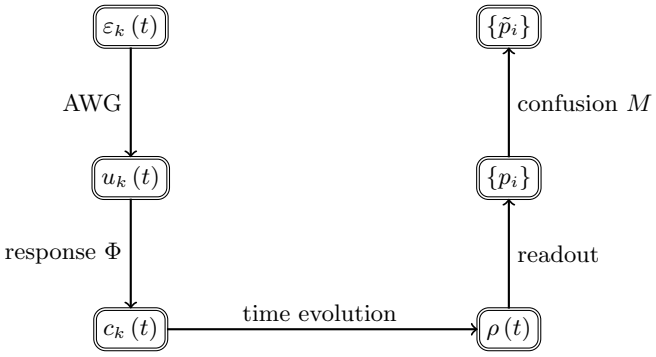
$$H(t) = H_0 + \sum_k c_k(t)\, H_k \ , \tag{10}$$

where the system is divided into an drift part H_0 and possibly several H_k that represent ways to steer system dynamics.

The dynamics of the system is given by solving the time-dependent equations of motion, be it of the Schrödinger



(a) A realistic control signal is modeled by starting from a sampled control function (red x) that is up-sampled and smoothed, according to Eq. (6).



(b) Readout is modelled by applying rescaling and transformations to the simulated populations $\tilde{p}_i = M p_i$.

FIG. 8: The process of emulating experimental procedure for signal processing and readout.

or Lindblad type, to compute the propagator

$$\begin{aligned} U(t) &= \mathcal{T} \exp\left(-\frac{\mathrm{i}}{\hbar} \int_{t_0}^{t} H(t')\, \mathrm{d}t'\right) \quad \text{or} \\ \Lambda(t) &= \mathcal{T} \exp\left(\int_{t_0}^{t} \mathcal{L}(t')\, \mathrm{d}t'\right) \ . \end{aligned} \tag{11}$$

The system is initialized in the state $\rho_i = |\phi_0\rangle\langle\phi_0|$, typically the ground state in the dressed basis. We allow for a state preparation error in the form of a finite initial temperature T resulting mixed state

$$\tilde{\rho}_i = \sum_k \frac{1}{Z} |\phi_k\rangle\langle\phi_k| \exp(-E_k/k_B T) \tag{12}$$

and $\{|\phi_k\rangle\}$ is the eigenbasis of H_0.

This yields the state of the system after applying the control pulse $\rho_f = U \rho_i U^\dagger$. On the final state ρ_f we now

apply a number of post processing steps emulating experimental procedure. From the state we obtain a vector of populations $\vec{p} = (p_k)$ by taking the absolute square of the diagonal. Measurement and classification error is modeled with a confusion matrix M such that the apparent populations are

$$\tilde{p}_i = M p_i \ . \tag{13}$$

In application, we will rarely perform a single gate or pulse in isolation. In general a pulse might be preceded by state preparation and followed by readout operations. Experiments like randomized benchmarking or the various flavors of tomography involve sending large pulse sequences to the experiment. Simulating these sequences as-is is not only inefficient, it also approaches the limit of classical computing hardware. We solve this problem in two steps:

First we define a gate set $\mathcal{G}$, where the elements $G \in \mathcal{G}$ might share control parameters, like amplitude, while others might be directly related, as rotations about x and y-axis for example have identical parameters, save for the angle $\alpha_{x,y}$ that need to be shifted by $\pi/2$. We put these parameters in a collective vector, taking care of the described redundancies and then proceed to compute each $G \in \mathcal{G}$ individually.

With the unitary representations of G on file, we compile them into the sequences that are to be evaluated. This allows us to obtain the goal function value for as many sequences as is desired, simply by multiplying the respective matrices. If local oscillator and qubit are not on resonance, one subtlety has to be taken care of in this compilation. In the lab frame, both the qubit and the drive signal rotate about the z-axis, which leads to a phase difference depending on gate time and qubit frequency. Therefore, the gates need to be interspersed with instantaneous z-rotations that align the rotating frames of the qubit and the local oscillator. A detailed description of this experimental procedure is layed out in REF Max.

B. Open-loop Model-based Control Optimization: C_1

In the typical setting of open-loop quantum optimal control [6, 7], given a model of a system, we search for the optimal control sequence (pulses) to drive the system to a desired state or generate a certain gate (unitary).

Control pulses may be subject to some ansatz (e.g. Gaussian pulse with DRAG correction [40]), or may be a direct piece-wise-constant parameterization of the waveform generator's digital-to-analog converter. In all cases, additional constraints may be imposed to conform with experimental feasibility, such as power and bandwidth limits. The optimal control task determines the choice of goal function (loss function) to be minimized, for instance average state infidelity for state transfer problem or unitary infidelity to implement quantum gates.

Formally, we parameterize the controls using a set $\bar{\alpha}$. Given a goal function $g(\bar{\alpha})$, we search for $\min_{\bar{\alpha}} g(\bar{\alpha})$. We note that the inevitable constraints on possible control fields induce the appearance of local minima in the optimization landscape - a problem which tends to worsen as we approach said constraints. Determining the value of $g(\bar{\alpha})$, necessitates simulating the dynamics of the quantum system, as represented by the model. Therefore the real-world applicability of the resulting pulses is limited by the accuracy of the model. And as Good Models are almost never used, the pulses resulting from optimal control are rarely directly usable, and must be tweaked using a closed-loop calibration [9].

Optimal control methods such as GRAPE [50], Krotov [51–54], and GOAT [8] have been devised to find minima of g in an efficient manner, i.e. by computing $\partial_{\bar{\alpha}} g(\bar{\alpha})$ and utilizing a gradient-based search. The derivation of these methods require that the system model follows a specific ansatz (usually $H(t) = H_0 + \sum_k c(\bar{\alpha}, t) H_k(t)$), which usually do not systimatically account for elements such as line response functions and SPAM error.

Automatic differentiation allows C_1 (and C_3) to compute the derivatives of a goal function with respect to the control (and system) parameters, allowing for an efficient gradient-based optimization. We note the use of automatic differentiation supplants the standard approach to optimal control, allowing very rich, and therefore potentially very accurate, models to be used.

C_1 makes use of L-BFGS [55] gradient-based optimizer, which is known for excellent performance for convex optimization landscapes, as it continuously estimates the goal function's Hessian. The disadvantage is its propensity to be trapped in local minima. We overcome the difficulty by using the hierarchy of models and control ansatz discussed in Sec. III. If this is insufficient, a short preliminary gradient-free search to find the convergence basin (see Sec. V C) most often resolves the problem.

C. Closed-loop, Model-free, Calibration: C_2

After the optimal control solution in respect to the current model is found via the C_1 step, it is then applied and refined in the closed-loop calibration, C_2. Gradients of the fidelity with respect to control parameters are no longer available, as we perform a *Black box Optimization* task.

In C_2, sets of control parameters $\bar{\alpha}_j$ are sent to the experimental setup, alongside instructions of how to evaluate the current controls. Protocols are usually taken from the standard calibration library - Rabi, Ramsey, ORBIT, RB, etc. The experiment returns the chosen figure of

merit for each set of control parameters, m_j. We note that this figure of merit contains information regarding the **entire** experimental setup - something which will be utilized in C_3.

While in simple experiments one might perform a manual or semi-automatic fine-tuning of each parameter, for more complex setups this way of calibration cannot be done without extraordinary effort [39]. The C^3 solution is to rely on sophisticated gradient-free optimization algorithms that automate this task.

Naturally, using an automated approach to solve the problem of calibration seems promising, and many experiments are currently using the *Nelder-Mead Method* [56, 57], for their calibration tasks. As we shall show below, this choice of algorithm is, most-often, ill-advised.

Black-box optimization of hyper-parameters is of great importance in the field of machine learning, in order to train the increasingly more complex neural networks. As a result, the machine learning community has invested significant resources searching for new and better algorithms. Therefore, we spent some time on investigating some of these new optimizers, as we suspect that big improvements can be made during the closed-loop calibration with smarter and more sophisticated algorithms than the ones that are currently used.

To this end we have made use of the Nevergrad package from Facebook Research [58] - a collection of gradient-free optimizer algorithms. The criteria against which we tested the optimizers include:

- **Noise resistance**: How well does an algorithm deal with noise? As experimental data is naturally not noise free.
- **Resistance to local extrema**: The parameter landscapes in the optimization tasks of the physical systems are usually containing not one global extremum but several local minima or maxima, as can be seen in Fig. 9. How likely is an algorithm to get stuck in such an extremum?
- **Dimension Scaling**
  As the number of qubits steadily increases so does the number of control parameters. On top of this, it has been shown that more complex pulses can increase the fidelity of operations on a quantum device [59]. Therefore, the property of scaling in dimension is important. How many parameters can an optimizer concurrently handle?
- **Convergence budget**: How long does an algorithm take to converge? If the algorithm takes too long, the automated calibration becomes useless.
- **Batching**: In many instances there is a fixed-time overhead associated with transferring new pulse shapes to the AWGs. To minimize overall optimization time, we would like to balance learning rate and total overhead by batching pulse evaluations.

After filtering 200 optimization methods with regards to the above mentioned criteria, we settled for the set of the following promising optimizer algorithms:

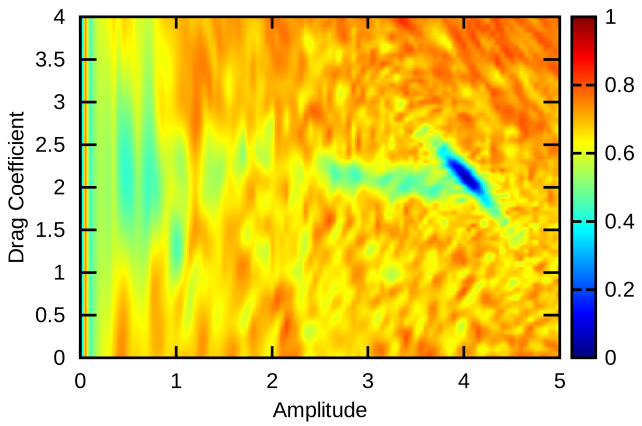
- Nelder-Mead Method [56, 57, 60]
- Powell's Method [61]
- (1+1) - Evolutionary Strategy [62, 63]
- Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) [47]

As the Nelder-Mead Method is well known and widely used, we decided to include it to show its performance as reference. The (1+1)-Evolutionary Strategy and the Covariance Matrix Adaption-Evolution Strategy (CMA-ES) are both algorithms of the more general group of $(\mu/\rho, \lambda)$-ES or $(\mu/\rho+\lambda)$-ES. Additionally, Powell's Method was included as its Nevergrad implementation showed promising results in the presence of local extrema.

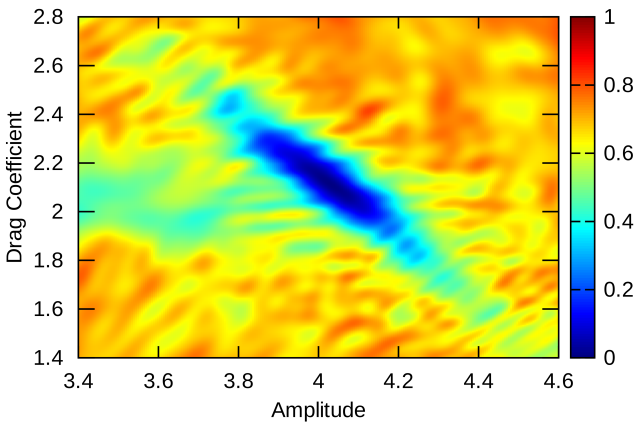
We opted to use a simulated physical system, in order to test in an environment as close to a real experiment as possible. In particular, we simulated a multi-level qubit on which we applied a single length RB-sequence. The control parameters $\bar{\alpha}_j$ were: the amplitude, the drag coefficient and the frequency offset of a drag corrected pulse used to control the qubit. The RB-sequence was chosen in a way that it ended in the ground state, and the simulation returned the infidelity of the overlap of the state after said sequence and the ground state as goal function. Fig. 9 shows the landscape of this goal function. An example of a calibration with the different optimizers on such a landscape can be seen in Fig. 10. The optimizers were all placed in a small area on the edge of the area of deep blue infidelities. On top of the intrinsic noise generated by using randomly generated RB-sequences, we added an additional 3% gaussian noise to the returned value to account for noise one would normally have in a real experiment. The immediate thing to notice is that all optimizers outperform the Nelder-Mead Method in the task at hand. In fact, the Nelder-Mead Method is stuck at the initial value. The brown line shows a modified CMA-ES version, with a more optimal choice of internal parameters for execution.

Given the results of our benchmarking efforts, we have come to the following conclusions:

- The Nelder-Mead Method is almost always the wrong choice. In almost all our benchmarking runs, it was outperformed by more sophisticated algorithms. Only in very simply cases, i.e. no noise or no local extrema did the Nelder-Mead Method perform well. In a typical experimental setup these conditions are rare.



(a) The landscape of the simulated system as a cut through the plane of amplitude and drag coefficient for a fixed RB-sequence. One can see a valley of local minima between a value of 2 and 2.5 for the drag coefficient.



(b) A higher resolution plot of the landscape of the deep blue minimum in the left plot. Local minima can be seen around the area of low infidelities.

FIG. 9: The landscape of the simulated goal function for a single RB-sequence. A random RB-sequence was generated that ended in the ground state. The control parameters $\bar{\alpha}_j$ were the amplitude, drag coefficient and frequency offset of a drag corrected pulse to control the qubit. The infidelity (seen as the color ranging from 0 to 1) was calculated from the overlap of the final state after the RB-sequence and the goal ground state. Fig. 9a shows the landscape of the simulated system as a cut through the plane of amplitude and drag coefficient. One can see a valley of local minima for a drag coefficient between 2 and 2.5 ending to the right in a deep minimum. Fig. 9b shows the area around this minimum in more detail.

- We recommend evolutionary strategies like CMA-ES or (1+1)-ES. These two algorithms did perform well in most cases and seem to be reliable against noise, can handle local extrema and are fast. Similar conclusions have been reached independently by [64].

- We furthermore think that sophisticated optimizer algorithms are needed to overcome the hurdles in the calibration of a many qubit device and therefore need a more in-depth analysis.

The investigation into new algorithms seems to be especially promising, as the field of machine learning in computer science is utilizing new algorithms and techniques that if adapted to the process of calibration bare the potential to increase the achieved fidelites in a much shorter time. Examples for such algorithms are the Adam algorithm in combination with Simultaneous Perturbation Stochastic Approximation (SPSA) or Bayesian Optimization.

At the end of the calibration process we have obtained a record of control parameters $\bar{\alpha}_j$ and their performance m_j which will provide the base for model learning.

D. Model Learning: C_3

Trying to extract the model from the data accumulated during calibration can thought of formally in analogy to the regular control problem, where instead the *model* parameters are optimized. For each measurement result g_j the corresponding control signal $\bar{\alpha}_j$ is applied to a model

$$H\left(\bar{\alpha}_j, \bar{\beta}, t\right) = H_0\left(\bar{\beta}\right) + \sum_k c_k\left(\bar{\alpha}_j, \bar{\beta}, t\right) H_k\left(\bar{\beta}\right) \quad (14)$$

and the model prediction for the measurement $m_j = m\left(\bar{\alpha}_j, \bar{\beta}\right)$ is calculated. From there the model matching goal function is defined as

$$f\left(\left\{\bar{\alpha}_j, g_j\right\}, \bar{\beta}\right) = \mathrm{Est}_j\left(m\left(\bar{\alpha}_j, \bar{\beta}\right), g_j\right) \quad (15)$$

with some estimation function Est that compares the data sets $\left\{m\left(\bar{\alpha}_j, \bar{\beta}\right)\right\}$ and $\{g_j\}$. Examples include the root mean square of the distance $\left|m\left(\bar{\alpha}_j, \bar{\beta}\right) - g_j\right|$ and logarithmic likelihood that both the simulation and experimental results are drawn from the same distribution.

As with C_1 the gradient $\partial_{\bar{\beta}} f$ is made available by automatic differentiation to facilitate convergence. The tendency of gradient based algorithms to get stuck in local minima can be avoided in a natural way: CMA-ES or a similar algorithm that uses batches or clouds of points to sample the landscape in the calibration step create a record that contains multiple candidates for initial points

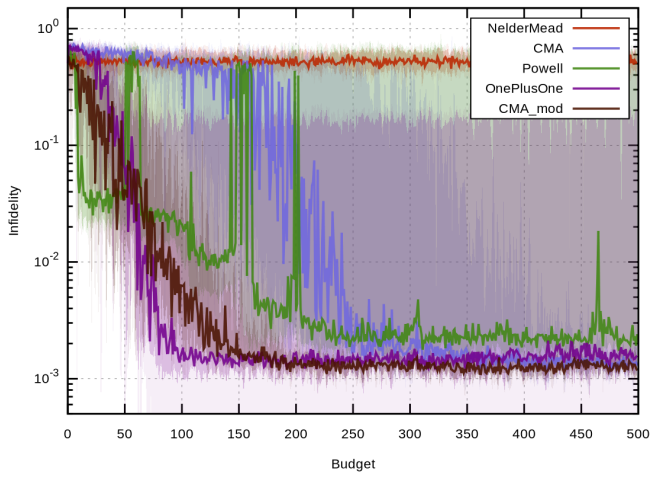


FIG. 10: Benchmarking Results of the optimizer algorithms Nelder-Mead Method, CMA-ES, Powell's Method, (1+1)-ES and a modified CMA-ES version with optimized internal algorithm parameters. The Nelder-Mead Method is outperformed by all other optimizers and stuck at the initial value. Within the first fifty evaluation steps the (1+1)-ES algorithm, the modified CMA-ES algorithm and Powell's Method algorithm do perform very well. The jumps in Powell's Method goal function value are due to an in-built restarting mechanism. The unmodified CMA-ES algorithm does need slightly more evaluations but in the end reaches a final infidelity of around 10^{-3}, much like the rest of the other algorithms(except the Nelder-Mead Method).

to feed to the gradient optimizer. The final best point of all of these optimizations gives the optimal value for the model parameters $\bar{\beta}$.

Where traditional experimental analysis requires a specific measurement to extract a specific quantity, e.g. spectroscopy reveals qubit frequency, Rabi oscillations shows the drive power and coupling strength between drive and system, Ramsay measurements are used to fine-tune resonances and estimate dephasing constant, the goal function formalism in C^3 provides a general measure for simulation performance. The values of experimental quantities are inherently encoded in the Good Model as long as the goal function is sensitive to them. From this point of view, the goal function formalism is an extension of the one-to-one relation between performed measurement and extracted quantity to multiple at the same time, thus it is able to handle correlations between parameters that would not be visible by sweeping one parameter at a time. In addition, it tells us something about the regime in which the model is valid. It predicts the experiment behavior exactly within the parameter space of the controls.

After converging C_3, the model predicts the experimental behavior up to the precision of the final goal value of $f\left(\bar{\beta}\right)$ and thus provides opportunities for analysis that wouldn't be accessible by merely calibrating. System dynamics can be investigated by looking at the simulated state at every time step, which makes it possible to identify coherent errors, like leakage out of the computational subspace or over rotations. The robustness of control pulses can be estimated by varying $\bar{\beta}$ around the optimal point and observing the change in the goal function to give an indication of the sensitivity to individual parameters.

E. Iterating models

Once C_3 has converged to a Good Model, additional analysis methods can be applied to the simulation:

- Sensitivity analysis for the model parameters. See Section **??**.
- Error budget for the model. See Section V F.
- Sensitivity analysis for C_1-optimized controls. Again, Section **??**.

However, due to the complexity of the physical systems, and the prevalence of local minima in all optimizations, it is unlikely that the first model and control ansatz attempted will result in the global optima (i.e. best possible pulses) for the physical system, and the associated model will be less than a Good Model. Therefore, we take the tried-and-tested experimental approach of starting with a simple model and simple control ansatz and iteratively refining it.

This can be accomplished in one of two ways: One may modify the model and repeat the C_3 fit, with either identical or different experimental (calibration) data, optionally retaining the optimized parameters which are shared by the previous and new models.

Alternatively, one may close the $C_1 \longrightarrow C_2 \longrightarrow C_3$ loop: The fitted model from the previous iteration is used for the C_1 control optimization, with a better model suggesting new control strategies and more sophisticated pulses.

The convergence speed and fidelity (final goal function value) of the subsequent C_2 calibration are mainly determined by how far the model is from the real experiment: If we have a Good Model, the results of the C_1 optimal control would perform as C_1 predicts, and an additional C_2 calibration will not improve things further.

We emphasize that at each of these steps the physicists' insights are required to evaluate optimization results and extend or discard models.

F. Error budgets via what-if

An error budget is an analysis of the infidelities observed in implemented gates, and associating parts of the infidelity to root causes (e.g: of the 0.004 infidelity, 0.001 is

caused by ZZ coupling to another qubit, 0.0025 is caused by Markovian noise and 0.0005 is due to the limited control bandwidth).

To determine the error model for our system, we first must determine a Good Modelfor the system. We recall that, by definite, an optimal-control C_1-type optimization performed on a Good Modelresults in pulses for which predicted fidelity and measured fidelity match.

To generate the error budget, the system model is modified to eliminate suspected possible causes of infidelity (e.g. T_1 is set to infinity), and control pulses are rederived using a C_1-type optimization. The gain in fidelity is error-budget line item value for the above modification of the system model.

VI. OPEN-SOURCE IMPLEMENTATION

The previously described functionality is implemented as an open source project at https://q-optimize.org under Apache 2 license. The software is written in Python to interface conveniently with common experiment controllers and has already been used in tandem with PycQED [65] and Labber. C^3 can interface at various levels of abstraction with an existing setup, from just sharing control parameters to fully taking care of signal generation, including the advanced modeling capabilities described in Sec. V A, and directly sending waveforms to signal generators.

To make use of C^3's calibration a function handle needs to be provided that takes either control fields or parameters and returns a result that is to be minimized. As long as this signature is maintained the function can be completely arbitrary.

Designed to be modular from the start, individual components of the C^3 code interact via accessible interfaces, allowing advanced users to make modifications to suit their needs, such as reordering the signal chain, including or excluding processing steps and parts of the model.

Each component of the control stack and model need to conform to a general boilerplate

```
def my_envelope_fuction(t, parameters):
        amplitude = parameters["Amp"]
        p2 = parameters["p2"]
        ...
return tf.some_math_function(amplitude, p2, t)
```

that specifies what control or model parameters it contains and how they are used. The only requirement to this code is that numerical functions have to be written in TensorFlow to allow for automatic differentiation.

User specified control functions need to provide a value at time t and can use a dictionary of parameters. Use of the TensorFlow package provides the gradient of this function to the C^3 code base without additional modification. For instance, the finite rise time of an AWG is realized with the following code:

```
class Response(Device):
    def __init__(..., rise_time, ...):
        self.params['rise_time'] = rise_time

    def process(self, iq_signal):
        ...
        t = self.params['rise_time'].get_value()
        sigma = t / 4
        ...
        # Convolution with a gaussian
        ...
        return self.signal
```

In this modular design, each class represents a physical device in the experiment that takes an input signal, applies some parameter-dependent function to it and returns the result. By putting the output of one device into the next a signal processing chain is represented.

At the core of C^3 is the physical simulator that is written to work with a general time-dependent Hamiltonian model to provide the physical state vector or density matrix after application of the control fields. In calculating figures of merit, the user can choose from a library of functions or also supply their own.

An example for an infidelity function is

```
def state_transfer_infid(U, psi_ideal, psi_0):
    psi_actual = tf.matmul(U, psi_0)
    overlap = tf_abs(
        tf.matmul(psi_ideal, psi_actual)
    )
    infid = 1 - overlap
    return infid
```

that evaluates the state transfer problem from $|\psi_0\rangle$ to $|\psi_{\text{ideal}}\rangle$. The propagator U is supplied by the simulation while TensorFlow computes the gradient of this function.

ACKNOWLEDGEMENTS

We acknowledge collaboration with ... as well as continuous collaboration with ... Some of this work is sponsored by the Intelligence Advanced Research Projects Activity (IARPA) through the LogiQ Grant No. W911NF-16-1-0114, by the European Union under OpenSuperQ and the ITN Qusco, and by the Germany ministry of science and education (BMBF) project VERTICONS.

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, Nature **574**, 505 (2019).

[2] J. Chow and J. Gambetta, "IBM Research Blog: Quantum Takes Flight: Moving from Laboratory Demonstrations to Building Systems," (2020), NoStop

[3] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, Physical Review A **100**, 032328 (2019).

[4] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O/'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, Nature **508**, 500 (2014).

[5] G. E. Box, Journal of the American Statistical Association **71**, 791 (1976).

[6] S. J. Glaser, U. Boscain, T. Calarco, C. P. Koch, W. Köckenberger, R. Kosloff, I. Kuprov, B. Luy, S. Schirmer, T. Schulte-Herbrüggen, D. Sugny, and F. K. Wilhelm, Eur. Phys. J. D **69** (2015).

[7] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, and T. Schulte-Herbrüggen, Phys. Rev. A **84**, 022305 (2011).

[8] S. Machnes, E. Assémat, D. Tannor, and F. K. Wilhelm, Phys. Rev. Lett. **120**, 150401 (2018).

[9] D. J. Egger and F. K. Wilhelm, Physical review letters **112**, 240503 (2014).

[10] P. Doria, T. Calarco, and S. Montangero, Phys. Rev. Lett. **106**, 190501 (2011).

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015), software available from tensorflow.org.

[12] B. Pokharel, N. Anand, B. Fortman, and D. A. Lidar, Physical review letters **121**, 220502 (2018).

[13] C. Ferrie, C. Granade, G. Paz-Silva, and H. M. Wiseman, New Journal of Physics **20**, 123005 (2018).

[14] A. Acharya, T. Kypraios, and M. Guţă, Journal of Physics A: Mathematical and Theoretical **52**, 234001 (2019).

[15] J. Helsen, F. Battistel, and B. M. Terhal, npj Quantum Information **5**, 1 (2019).

[16] J. Y. Sim, J. Shang, H. K. Ng, and B.-G. Englert, Physical Review A **100**, 022333 (2019).

[17] L. Govia, G. Ribeill, D. Ristè, M. Ware, and H. Krovi, Nature Communications **11**, 1 (2020).

[18] Z. Hou, J.-F. Tang, C. Ferrie, G.-Y. Xiang, C.-F. Li, and G.-C. Guo, Physical Review A **101**, 022317 (2020).

[19] S. Kimmel, M. P. da Silva, C. A. Ryan, B. R. Johnson, and T. Ohki, Physical Review X **4**, 011050 (2014).

[20] Y. Gazit, H. K. Ng, and J. Suzuki, Physical Review A **100**, 012350 (2019).

[21] M. Rol, L. Ciorciaro, F. Malinowski, B. Tarasinski, R. Sagastizabal, C. Bultink, Y. Salathe, N. Haandbaek, J. Sedivy, and L. DiCarlo, Applied Physics Letters **116**, 054001 (2020).

[22] J. P. van Dijk, E. Kawakami, R. N. Schouten, M. Veldhorst, L. M. Vandersypen, M. Babaie, E. Charbon, and F. Sebastiano, Physical Review Applied **12**, 044054 (2019).

[23] M. Jerger, A. Kulikov, Z. Vasselin, and A. Fedorov, Physical review letters **123**, 150501 (2019).

[24] I. N. Hincks, C. E. Granade, T. W. Borneman, and D. G. Cory, Phys. Rev. Applied **4**, 024012 (2015).

[25] I. Hincks, C. Granade, T. Borneman, and D. Cory, Physical Review Applied **4**, 024012 (2015).

[26] S. Gustavsson, O. Zwier, J. Bylander, F. Yan, F. Yoshihara, Y. Nakamura, T. P. Orlando, and W. D. Oliver, Physical review letters **110**, 040502 (2013).

[27] R. S. Gupta, A. R. Milne, C. L. Edmunds, C. Hempel, and M. J. Biercuk, arXiv preprint arXiv:1904.07225 (2019).

[28] R. Harper, S. T. Flammia, and J. J. Wallman, arXiv preprint arXiv:1907.13022 (2019).

[29] J. Kelly, R. Barends, A. Fowler, A. Megrant, E. Jeffrey, T. White, D. Sank, J. Mutus, B. Campbell, Y. Chen, *et al.*, Physical Review A **94**, 032321 (2016).

[30] L. Casparis, T. Larsen, M. Olsen, F. Kuemmeth, P. Krogstrup, J. Nygård, K. Petersson, and C. Marcus, Physical review letters **116**, 150505 (2016).

[31] C. Miquel, J. P. Paz, and W. H. Zurek, Physical review letters **78**, 3971 (1997).

[32] R. C. Bialczak, R. McDermott, M. Ansmann, M. Hofheinz, N. Katz, E. Lucero, M. Neeley, A. OConnell, H. Wang, A. Cleland, *et al.*, Physical review letters **99**, 187006 (2007).

[33] J. J. Burnett, A. Bengtsson, M. Scigliuzzo, D. Niepce, M. Kudra, P. Delsing, and J. Bylander, npj Quantum Information **5**, 1 (2019).

[34] M. Y. Niu, V. Smelyanskyi, P. Klimov, S. Boixo, R. Barends, J. Kelly, Y. Chen, K. Arya, B. Burkett, D. Bacon, *et al.*, arXiv preprint arXiv:1912.04368 (2019).

[35] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, Phys. Rev. A **77**, 012307 (2008).

[36] A. Zhang, J. Xie, H. Xu, K. Zheng, H. Zhang, Y.-T. Poon, V. Vedral, and L. Zhang, Physical Review Letters **124**, 040402 (2020).

[37] P. Mundada, G. Zhang, T. Hazard, and A. Houck, Physical Review Applied **12**, 054023 (2019).

[38] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, arXiv preprint arXiv:2001.02826 (2020).

[39] J. Kelly, P. O'Malley, M. Neeley, H. Neven, and J. M. Martinis, arXiv preprint arXiv:1803.03226 (2018).

[40] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm, Phys. Rev. Lett. **103**, 110501 (2009).

[41] M. GeversÃĂ, European journal of control **11**, 1 (2005).

[42] C. R. Rojas, J. S. Welsh, G. C. Goodwin, and A. Feuer, Automatica **43**, 993 (2007).

[43] L. Ljung, Annual Reviews in Control **34**, 1 (2010).

[44] X. Bombois, M. Gevers, R. Hildebrand, and G. Solari, Communications in Information and Systems **11**, 197 (2011).

[45] J. Kelly, R. Barends, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, I.-C. Hoi, E. Jef-

frey, A. Megrant, J. Mutus, C. Neill, P. J. J. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. C. White, A. N. Cleland, and J. M. Martinis, Phys. Rev. Lett. **112**, 240504 (2014).
[46] E. Magesan, J. M. Gambetta, B. R. Johnson, C. A. Ryan, J. M. Chow, S. T. Merkel, M. P. da Silva, G. A. Keefe, M. B. Rothwell, T. A. Ohki, M. B. Ketchen, and M. Steffen, Phys. Rev. Lett. **109**, 080505 (2012).
[47] N. Hansen, S. D. Müller, and P. Koumoutsakos, Evolutionary Computation **11**, 1 (2003).
[48] M. Werninghaus, D. J. Egger, F. Roy, S. Machnes, F. K. Wilhelm, and S. Filipp, "Leakage reduction in fast superconducting qubit gates via optimal control," (2020).
[49] D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta, Physical Review A **96**, 022330 (2017).
[50] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, Journal of Magnetic Resonance **172**, 296 (2005).
[51] R. Kosloff, S. Rice, P. Gaspard, S. Tersigni, and D. Tannor, Chem. Phys. **139**, 201 (1989).
[52] A. P. Peirce, Mohammed, A. Dahleh, and H. Rabitz, Phys. Rev. A **37**, 4950 (1988).
[53] D. M. Reich, M. Ndong, and C. P. Koch, The Journal of Chemical Physics **136**, 104103 (2012).
[54] M. H. Goerz, D. Basilewitsch, F. Gago-Encinas, M. G. Krauss, K. P. Horn, D. M. Reich, and C. P. Koch, SciPost Phys. **7**, 80 (2019).
[55] J. Nocedal, Mathematics of Computation **35**, 773 (1980).
[56] J. A. Nelder and R. Mead, The Computer Journal **7**, 308 (1965).
[57] L. Han and M. Neumann, Optimization Methods and Software **21**, 1 (2006).
[58] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform," https://GitHub.com/FacebookResearch/Nevergrad (2018).
[59] S. Kirchhoff, T. Keßler, P. J. Liebermann, E. Assémat, S. Machnes, F. Motzoi, and F. K. Wilhelm, Phys. Rev. A **97**, 042348 (2018).
[60] J. Nelder and R. Mead, Comput. J. **7**, 308 (1965).
[61] J. G. Powles, P. Phys. Soc. **71**, 497 (1958).
[62] P. A. Borisovsky and A. V. Eremeev, in *FOGA* (2002) pp. 271–288.
[63] S. Droste, T. Jansen, and I. Wegener, Theoretical Computer Science **276**, 51 (2002).
[64] J. Rapin, M. Gallagher, P. Kerschke, M. Preuss, and O. Teytaud, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO 19 (Association for Computing Machinery, New York, NY, USA, 2019) p. 18881896.
[65] .

Appendix A: Details on synthetic model matching

Appendix B: Examples for configuration files

1. Open loop optimal control

A json configuration file for C_1 optimal control for the synthetic example in section IV. The nested list in the opt_map signifies that all four gates share the same three parameters. The optimization problem therefore has three parameters instead of 12.

```
{
  "dir_path" : "/localdisk/c3logs/",
  "algorithm" : "lbfgs",
  "fid_func" : "lind_average_infid",
  "gateset_opt_map" : [
    [
      ["X90p", "d1", "gauss", "amp"],
      ["Y90p", "d1", "gauss", "amp"],
      ["X90m", "d1", "gauss", "amp"],
      ["Y90m", "d1", "gauss", "amp"]
    ],
    [
      ["X90p", "d1", "gauss", "freq_offset"],
      ["Y90p", "d1", "gauss", "freq_offset"],
      ["X90m", "d1", "gauss", "freq_offset"],
      ["Y90m", "d1", "gauss", "freq_offset"]
    ],
    [
      ["X90p", "d1", "gauss", "delta"],
      ["Y90p", "d1", "gauss", "delta"],
      ["X90m", "d1", "gauss", "delta"],
      ["Y90m", "d1", "gauss", "delta"]
    ]
  ]
}
```

2. Model learning

```
{
  "dir_path" : "/localdisk/c3logs/",
  "algorithm" : "lbfgs",
  "sampling" : "even",
  "batch_size" : 20,
  "estimator" : "rms",
  "callback_est" : [
    "median",
    "rms",
    "stds",
    "gauss",
    "binom",
    "rms_stds",
    "std_diffs"
  ],
  "callback_figs" : [
    "exp_vs_sim",
    "exp_vs_sim_2d_hist"
  ],
  "exp_opt_map" : [
    ["init_ground", "init_temp"],
    ["Q1", "freq"],
    ["Q1", "anhar"],
    ["Q1", "t1"]
  ]
}
```