



Optimizing quantum control pulses with complex constraints and few variables through autodifferentiation

Yao Song ¹, Junning Li,² Yong-Ju Hai,^{1,3} Qihao Guo,¹ and Xiu-Hao Deng ^{1,4,*}

¹*Shenzhen Institute of Quantum Science and Engineering,*

Southern University of Science and Technology, Shenzhen, Guangdong 518055, China

²*Department of Physics, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR, China*

³*Department of Physics, Southern University of Science and Technology, Shenzhen 518055, China*

⁴*Guangdong Provincial Key Laboratory of Quantum Science and Engineering,*

Southern University of Science and Technology, Shenzhen, Guangdong 518055, China



(Received 11 October 2021; accepted 5 January 2022; published 31 January 2022)

Applying optimal control algorithms on realistic quantum systems confronts two key challenges: to efficiently adopt physical constraints in the optimization and to minimize the variables for the convenience of experimental tuneups. To resolve these issues, we propose an algorithm by incorporating multiple constraints into the gradient optimization over piecewise pulse constant values, which are transformed to contained numbers of the finite Fourier basis for bandwidth control. Such complex constraints and variable transformation involved in the optimization introduce extreme difficulty in calculating gradients. We resolve this issue efficiently utilizing autodifferentiation on the machine learning platform TensorFlow. We test our algorithm by finding smooth control pulses to implement single-qubit and two-qubit gates for superconducting transmon qubits with always-on interaction, which remains a challenge of quantum control in various qubit systems. Our algorithm provides a promising optimal quantum control approach that is friendly to complex and optional physical constraints.

DOI: [10.1103/PhysRevA.105.012616](https://doi.org/10.1103/PhysRevA.105.012616)

I. INTRODUCTION

Potential ground-breaking quantum technologies, such as quantum computing, quantum sensing, and quantum metrology [1–3], become more and more feasible with the tremendous progress of quantum control technology on various physical systems [4–7]. Based on growing knowledge about quantum systems interacting with the environment, multifarious approaches have been developed to improve the control precision [8–14]. Even so, further optimizing quantum control still highly relies on numerical approaches [14–20]. Practical quantum optimal control (QOC) [21–25] should satisfy the requirements and constraints in the physical systems, such as more realistic Hamiltonians, maximal field strengths, finite sampling rates, limited bandwidths [26–29], etc. Also, for efficient calibration in experiments, the control field waveform should depend on as few variables as possible. Other than the physical considerations, the optimization algorithm ought to be fast and accurate and should be extensible to larger systems. As one of the most successful numerical optimization algorithms, GRAPE has been applied to many physical systems, including NMR qubits [15,30–33], superconducting qubits in three-dimensional cavities [34–38], nitrogen-vacancy (NV) centers in diamond [5,39,40], etc. However, adapting GRAPE to multiple realistic constraints remains challenging. Different from GRAPE, another numerical algorithm, CRAB, has been proposed to generate smooth

control waveforms, towards applications in cold atoms [41]. There are also many other proposed algorithms leading to promising applications [17,18,42,43].

It's worth noting that quantum optimal control is a non-convex optimization problem [44–46], but we can still find a good local minimum with some strategies [46,47]. Depending on how to parametrize the control field's variables, QOC algorithms could be assorted into two classes: 1. *Piecewise constant* (PWC) discretizes the field pulse as a sequence of piecewise-constant field strengths and the value of each piece as the optimization variable [15,16,42,48,49]. Varying each variable causes local variation of the whole waveform and hence the local variation of the expectation function, which could help with fine searching within a local minimum. And then gradient-based optimization could efficiently proceed [15,17,18,42]. A tradeoff arises between inaccurate PWC dynamics for finite discretization rates and the cost of computing the gradients versus massive variables. Also, PWC waveforms are not smooth and could easily contain fast fluctuations. Filtering the optimized waveform in the postoptimization deforms the output pulses from optimum. 2. *Chopped basis* (CB) optimization uses parameters related to a finite set of basis expanding the control waveform. The basis is usually analytic functions such as Gaussian, tangential, sinusoidal, or Fourier functions [14,19,50], etc., which easily guarantees the smoothness of the optimized waveform. However, adding the physical constraints deforms the processed pulses in an unknown way [14,19], so the output pulses lose analyticity. Furthermore, a small change of one expansion coefficient leads to global deformation of the pulse waveform, and also

*dengxh@sustech.edu.cn

coefficients of different bases have different effects. Therefore the nonfine searching of CB optimization could sometimes jump out a local minimum and get a relatively bad result compared to the PWC method. Alternatively, constraints could be incorporated into optimization with Lagrange multipliers, but the calculation of the analytical differentiation brings in another source of complexity and difficulty [43,44,51].

Here in this article, we propose an optimization algorithm, the *complex and optional constraints optimization with autodifferentiation* (COCOA), to tackle the problems summarized above. The algorithm parametrizes the control waveform based on a truncated Fourier series, while the optimization performs the gradient in the convex landscape of PWC parameter space. The transformation between the aforementioned two parametrization systems is bridged via approximate discrete Fourier transformation (DFT) and inverse discrete Fourier transformation (iDFT). The advantage of combining these two parametrization systems is that all the pulse constraints can be incorporated in the optimization process instead of postoptimization. The application of DFT and iDFT within the optimization iteration introduces further physical advantages: 1. hard bounds on the bandwidth by limiting the Fourier basis; 2. definite expansion with preselected basis; 3. analytical expression of output pulses with a small number of parameters for tuning; and 4. convenient application of pulse predistortion because the transfer and filter functions are in the frequency domain [28,52,53]. However, embedding these constraints and the transformation of parametrization systems into the optimization iteration raises extreme difficulty in solving the analytical gradient equations. We resolve this issue using autodifferentiation during the one-round backpropagation process provided by TensorFlow [54]. With TensorFlow, the automatic calculation of gradient could be further extended to various physical systems. This paper explains in detail how COCOA works and presents some numerical results to demonstrate the efficiency and its advantage compared with GRAPE and CRAB.

This paper is organized as follows. First, we take an overview of quantum optimal control theories and numerical methods. In Sec. III we present the algorithmic description of COCOA and explain how it combines the advantages of two categories of QOC approaches and performs an optimization task under complex and optional constraints. Then in Sec. IV, we apply COCOA to find optimal quantum-gate pulses for superconducting qubits with always-on interactions. We simulate two models: cavity-mediated two-qubit systems and direct capacitively coupled systems. We demonstrate COCOA's advantages by comparing with the representative algorithms GRAPE and CRAB of the two QOC categories. In Sec. V, we conclude this paper.

II. QUANTUM OPTIMAL CONTROL

A. General formalism

We consider a general Hamiltonian in the QOC problem as

$$H(t) = H_d + H_c(t), \quad (1)$$

where H_d is the drift Hamiltonian of the system. $H_c(t)$ is the time-dependent Hamiltonian which is to be optimized to control the quantum system to undergo a desired time evolution.

The dynamics of the system steered by the total Hamiltonian in Eq. (1) satisfied the Schrödinger equation $|\dot{\psi}(t)\rangle = -iH(t)|\psi(t)\rangle$, with the time evolution operator satisfying $\dot{U}(t) = -iH(t)U(t)$, or an integral form $U(t) = \mathcal{T}e^{-i\int H(t)dt}$, where \mathcal{T} is the time-ordering operator. A generic form of the control Hamiltonian is $H_c(t) = H_c(\{\Omega_j(t)\}, t)$, where $\{\Omega_j(t)\}$ are a set of time-dependent control pulses to be optimized. In our examples presented later, $\{\Omega_j(t)\}$ are chosen to be the envelopes of microwave drives on qubits, where $j = 1x, 1y, 2x, 2y$, i.e., $1x$ means rotating around the x axis for qubit 1. For different QOC problems, the expectation function could be customized. A constrained optimization could be performed by combining penalty functions into the expectation function with Lagrange multipliers [43,51]. In the specific examples discussed in this article, we study the performance of a quantum gate at final time $t = T$ as an average over all possible initial quantum states. It can be quantified by the average gate fidelity [55], defined as

$$f = \frac{1}{d(d+1)} [\text{Tr}(MM^\dagger) + |\text{Tr}(M)|^2], \quad (2)$$

where $M = U_{tar}U_T$, and d is the dimension of the quantum system. U_{tar} denotes the target gate that you want to optimize. Therefore we use infidelity as the cost function to be minimized, i.e., $F = 1 - f$.

B. Realistic requirements

Traditional pulse optimization algorithms are confronted with many issues while applying to realistic systems. We summarize various realistic issues below and design the COCOA optimization to bridge the gap between numerical optimization and experimental applications.

1. *Pulse predistortion.* Pulse distortion as one of the major issues could take place in the following process [56]: (i) Pulse generation with finite sampling rates, which could be modeled as a finite impulse response (FIR) filter. (ii) Transmission of signal, where an infinite impulse response (IIR) transfer function could be used to model the distortion. (iii) Numerical distortion when postprocessing the optimal pulse for purposes such as smoothening, which could be modeled as the postoptimization filter function. Commonly, the processed pulse is not optimal anymore. Pulse predistortion with inverse filter function could be applied in experiments to compensate for these effects [28,57]. Since the filter functions are in the frequency domain, it would be much more convenient for predistortion if the numerical output pulse is an analytical function for frequency rather than in the PWC form.

2. *Pulse constraint.* Optimal pulses should satisfy various physical constraints, such as smoothness, finite bandwidth, bounded amplitude, starting and ending at some designated values, robustness to some randomness (e.g., noises), and so on. Postprocessing the optimized pulses with constraints results in the numerical pulse distortion mentioned above. So it is necessary to incorporate constraints into the optimization. However, efficiently adding pulse constraints to the optimization is challenging because the gradient might be too complicated to compute. Traditionally, conditional expressions such as the if-else paragraph could be added into the optimization solver. But this results in possible loopholes and

Algorithm 1. Complex and optional constraints optimization (COCOA) for pulse engineering.

Require: Cost function: F ; initial pulse sequence: $\Omega_j^0(t)$; N_c ; α ; iterations: R ; ϵ_0 ; ϵ_1

Ensure: Optimized pulse: $\Omega_j^*(t)$

1: Discretization: $\Omega_j^0(t) \rightarrow \{\Omega_j^0[k]\}$;

2: **Repeat**

3: Record computational graph for autodifferentiation;

4: Pulse constraint: $\{\Omega_j^r[k]\} \rightarrow \{\tilde{\Omega}_j^r[k]\}$;

5: Bandwidth control: $\{\tilde{\Omega}_j^r[k]\} \rightarrow \{\hat{\Omega}_j^r[k]\}$;

6: Evolution: $U_k = e^{-i(H_d + \sum_j \hat{\Omega}_j^r[k] H_j |t_k|) \Delta t}$, $U_T = U_N U_{N-1} \dots U_1$;

7: Cost function: $F(U_T, S_T)$;

8: Calculate gradient using autodifferentiation: $\{\frac{\partial F}{\partial \Omega_j^r[k]}\}$;

9: Update control pulse sequence using SGD or Adam algorithm.

10: **Until** $[F(U_T, S_T) < \epsilon_0$ or $\|\frac{\partial F}{\partial \Omega_j^r[k]}\| < \epsilon_1]$.

low efficiency. So a better approach is still open for investigation.

3. *Pulse parameters.* Optimizing the pulse parameters in experiments is necessary and could become exponentially strenuous when the number of parameters grows. Therefore, it is desired to obtain optimal pulses with few parameters for experimental tuneups.

4. *Analyticity of the pulse.* The analyticity is defined as an explicit and analytical function or a definite summation of several analytical functions, which helps generate and tune control pulses in experiments.

In this article we will show how the COCOA algorithm is engineered to satisfy the above requirements efficiently. And we will test its performance via realistic optimization tasks.

III. COCOA ALGORITHM

A. Pseudocode

The pseudocode of the COCOA algorithm can be seen in Algorithm 1. Here α denotes the learning rate, ϵ_0 and ϵ_1 are the threshold of cost function and the norm of the gradient, and k is the index of the discrete-time sequence.

B. Algorithm settings

The process of COCOA is shown as a flowchart in Fig. 1. The four main nodes in COCOA are pulse constraint, bandwidth control, evolution, and cost function. Pulse constraints and bandwidth control nodes are the core nodes, which always do a pretreatment on the pulse before evolution, while the form of evolution node and cost function node depends on the problem and optimization task. In the following we elaborate on each node and illustrate several unique features of COCOA.

1. Ansatz for pulses

COCOA could use different settings of initial pulses with arbitrary waveform, including random guesses of PWC sequence, random combinations of Fourier series, and specific analytical forms based on prior knowledge. The efficiency remains at a high level for various settings, as will be demonstrated later in the manuscript and shown in Figs. 3, 6, 10, respectively. In this section we focus on the illustration COCOA could use different settings of initial

pulses with arbitrary waveform, including random guesses of PWC sequence, random combinations of Fourier series, and specific analytical forms based on prior knowledge. The efficiency remains at a high level for various settings, as will be demonstrated later in the manuscript and shown in Figs. 3, 6, 10, respectively. In this section we focus on the illustration of processing pulses in the algorithm. Without loss of generality, the initial pulses are set to be the truncated Fourier basis function in the algorithm.

The control pulse can always be written as an expression with the parameters $\Omega_j(\vec{p}, t)$, where \vec{p} is the parameter vector. According to the actual demand, the analytical pulse form can be any form. For the consideration of limiting pulse bandwidth, without loss of generality we take the chopped Fourier basis functions as

$$\Omega_j(\vec{p}, t) = a_0 + \sum_{n=1}^{N_c} A_{jn} \cos(\omega_{jn} t) + B_{jn} \sin(\omega_{jn} t), \quad (3)$$

where the pulse parameter set \vec{p} is formed with Fourier expansion parameters

$$\vec{p} = \{A_{jn}, B_{jn}, a_0\}. \quad (4)$$

Analytical pulse functions solved from different theories could be exactly or approximately transformed to the chopped Fourier basis for optimization, such as Slepian pulses [58], SWIPHT pulses [9,10], geometric pulses [12], and so on. We demonstrate this in Sec. IV B 1. Note that filter functions for pulse predistortion could be directly applied on the Fourier basis, which brings added convenience to experimental tuneups.

In the PWC optimization, $\Omega_j(\vec{p}, t)$ is discretized to an N -length sequence with sampling frequency $f_s = N/T$, where T is the total gate time:

$$\Omega_j(\vec{p}, t) \xrightarrow{\text{sample}} \{\Omega_j[\vec{p}, k]\}, k = 1, \dots, N. \quad (5)$$

For convenience, the discretized temporal sequence of the PWC ansatz is denoted as $\{\Omega_j[k]\}$, $k = 1, \dots, N$.

2. Amplitude constraint

A realistic system limits the maximal strength of the control field. Also, a single control pulse starts and ends at zero. As a traditional way in textbook [15,59], this constraint enters

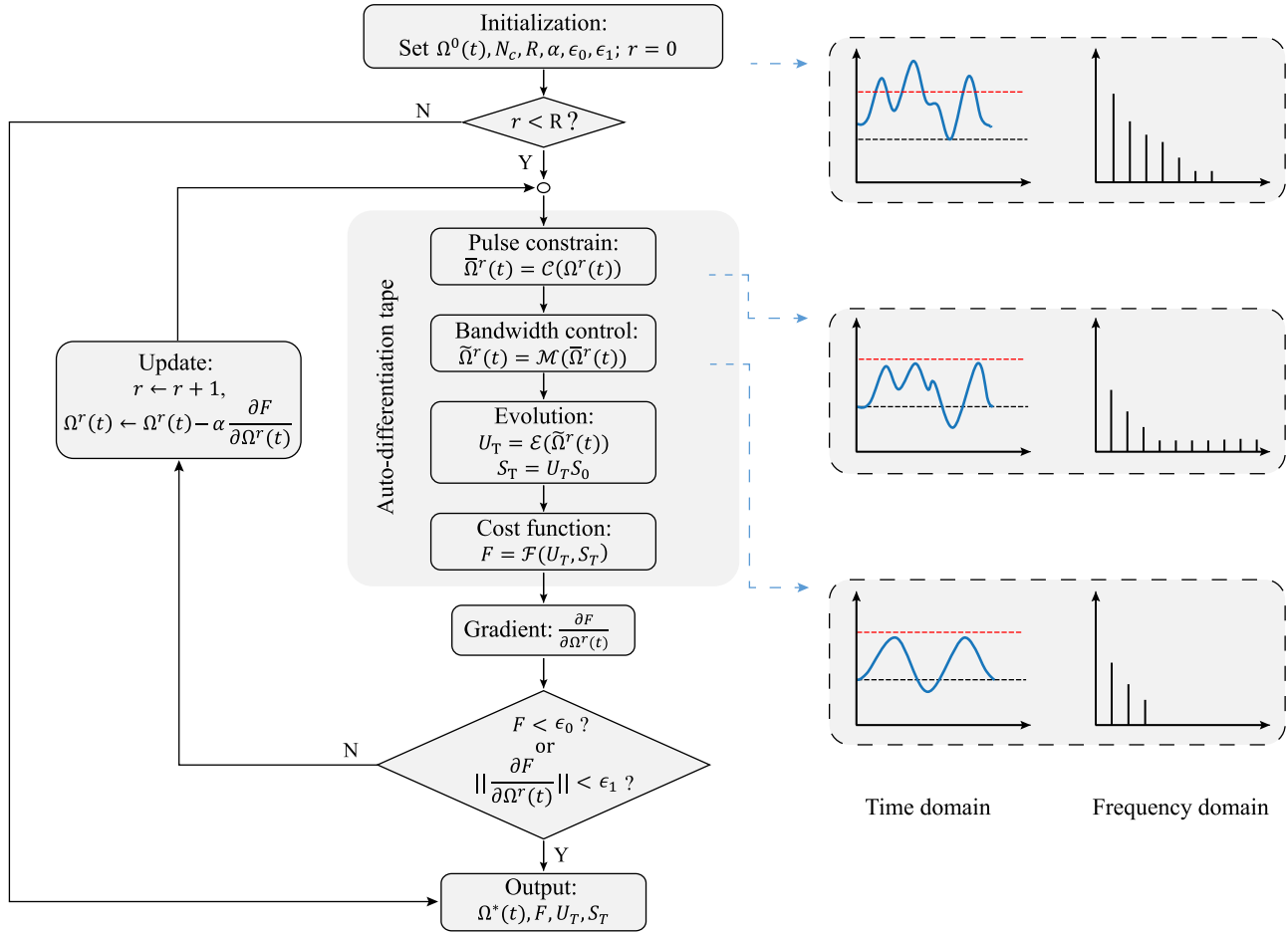


FIG. 1. Flow diagram of COCOA. We use the transform functions \mathcal{C} , \mathcal{M} , \mathcal{E} , \mathcal{F} to denote the four main nodes. The autodifferentiation tape is just a recorder that records all the computing processes automatically by TensorFlow and will be used for the computing gradient. The figures on the left show how the pulse changes after each specific node in the time and frequency domain. S_0 and U_T denote the initial state and the final operator, respectively.

the optimization cost function by adding up with the control power defined as $J = \lambda \int_0^T \Omega_j^2(t) dt$, where the weight $\lambda > 0$. However, the fidelity of the optimized pulse will be

lower with this term added. Furthermore, this way just gives a soft constraint on the amplitude maximum, which could be harmful when physical systems have a hard limit on control

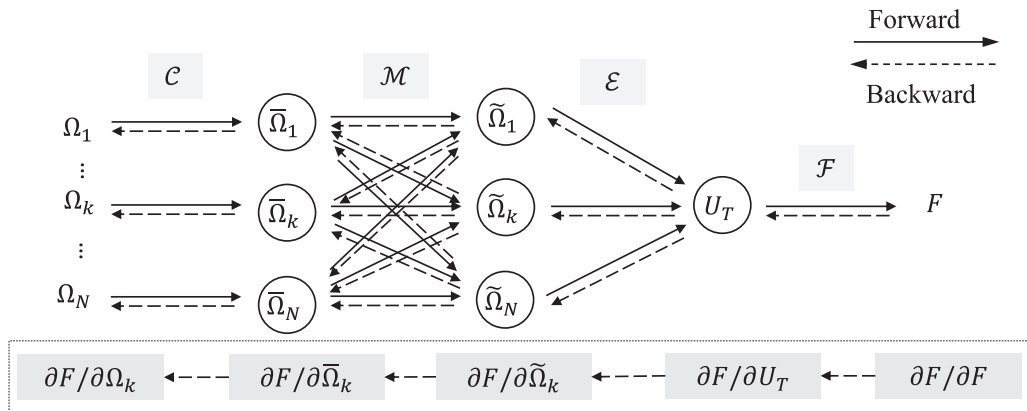


FIG. 2. Computational graph of autodifferentiation. \mathcal{C} , \mathcal{M} , \mathcal{E} , \mathcal{F} denote four transform layers, corresponding to each node. $\bar{\Omega}_k, \tilde{\Omega}_k, U_T, F$ are the output sequence after each transform layer. The solid line (dashed line) is the forward propagation (backpropagation). The subscript k denotes the index of the control sequence Ω . The dashed box at the bottom shows how autodifferentiation computes the gradient of F with respect to Ω_k , following the chain rule.

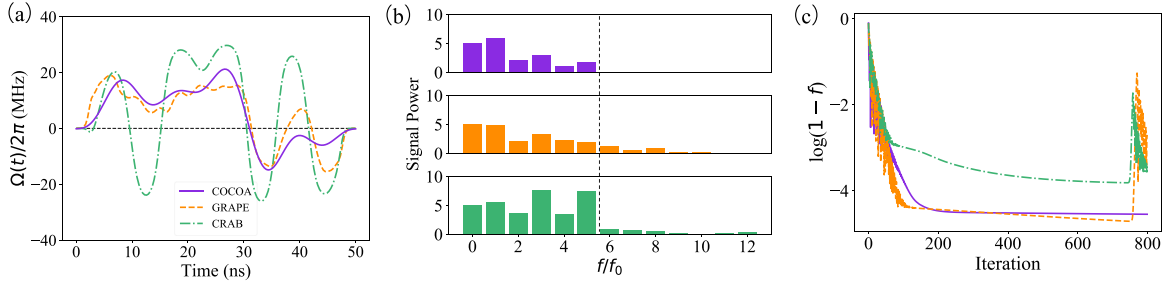


FIG. 3. Algorithm comparison. (a) Optimized pulse with zero amplitude at $t = 0$ ns and $t = 50$ ns. The pulse amplitude is limited between $[-30, 30]$ MHz. (b) Frequency spectrum of each pulse. $f_0 = 1/T = 0.02$ GHz. As we can see, GRAPE [orange dashed line in (a)] and CRAB [green dash-dotted line in (a)] have more high-frequency components induced by pulse constraints or by the algorithm itself. The gray dashed line shows the truncated components of the COCOA algorithm. (c) Gate infidelity changing vs iteration, showing the efficiency of each algorithm.

amplitude. In our algorithm a strong constraint to the pulse amplitude is added by passing the control pulses through a sigmoid window function, similar to GOAT [14]:

$$S_{\text{down}}(t, g) = \frac{1}{1 + e^{gt}}, \quad (6)$$

$$S_{\text{up}}(t, g) = 1 - S_{\text{down}}(t, g), \quad (7)$$

$$S_{\text{amp}}(\Omega_j[k], l, u) = \left[2S_{\text{up}}\left(\frac{\Omega_j[k] - \frac{u+l}{2}}{\frac{u-l}{2}}, 0.5\right) - 1 \right] \frac{u-l}{2} + \frac{u+l}{2}, \quad (8)$$

where g is the ascent or descent gradient of the window function, and l, u are the lower and upper bounds of the pulse amplitude.

The total amplitude constraint transformation reads

$$\begin{aligned} & \mathcal{C}(\Omega_j[k], t, l, u, g, \Delta t) \\ &= S_{\text{up}}\left(\frac{t - \Delta t}{T}, g\right) S_{\text{down}}\left(\frac{t - (T - \Delta t)}{T}, g\right) S_{\text{amp}}(\Omega_j[k], l, u), \end{aligned} \quad (9)$$

where Δt is the width of the ascent (descent) edge. The first and the second sigmoid function ensures zero amplitude at $t = 0$ and $t = T$, and thus satisfies the second physical constraint. The last one bounds the amplitude to the $[l, u]$ range.

3. Bandwidth control

In this node we modulate the control pulse to a bandwidth-limited one in the frequency domain. First we transform the pulse sequence from the time domain into the frequency domain using discrete Fourier transform (DFT):

$$X[n] = \sum_{k=1}^N \Omega_j[k] e^{i(2\pi n/N)k}. \quad (10)$$

After DFT, We get a complex sequence $\{X[n]\}$, $n = 1, \dots, N$. Assuming the upper cut-off frequency is f_{th} , we can derive the maximal Fourier component number N_c ,

$$N_c \leq \text{Int}\left(N \frac{f_{th}}{f_s}\right), \quad (11)$$

where $\text{Int}(\cdot)$ indicates rounding down. Here N_c is a hyperparameter in our algorithm and it affects the pulse's simplicity, smoothness, and numerical accuracy. We discuss it in detail in Sec. IV.

Then the complex sequence in the frequency domain becomes

$$Y[n] = \begin{cases} \tilde{X}[n], & n \in [1, N_c] \cup [N - N_c, N] \\ 0, & n \in [N_c + 1, N - N_c - 1] \end{cases}. \quad (12)$$

After DFT, the higher Fourier components over N_c , namely, the complex sequence elements from $X[N_c + 1]$ to $X[N - N_c - 1]$, will be set to zero, i.e., limiting the bandwidth. Because the complex sequence after DFT is conjugate symmetric, i.e., $X[n] = X^*[N - n]$, and the n th spectral component is contained in $X[n]$ and $X[N - n]$, we have to set both of them to zero to do a bandwidth limitation. For more details, you can refer to Appendix A.

Then we apply the inverse transformation of DFT, called iDFT, to transform the pulse back to the time domain,

$$\tilde{\Omega}_j[k] = \frac{1}{N} \sum_{n=1}^N Y[n] e^{i(2\pi k/N)n}. \quad (13)$$

After iDFT, the pulse sequence $\{\tilde{\Omega}_j[t_k]\}$ is a smooth pulse sequence with limited bandwidth. Its functional form in continuous time domain is denoted as

$$\tilde{\Omega}_j(t) = a_0 + \sum_{n=1}^{N_c} A_n \cos\left(n \frac{2\pi f_s}{N} t + \phi_n\right). \quad (14)$$

It is worth noting that this is the functional form of our final optimized waveform, which is a finite Fourier basis function. More details about DFT and iDFT can be seen in Appendix A.

4. Evolution

For the evolution node, the smooth, analytical, and bandwidth-limited control pulse, Eq. (14), obtained from the previous nodes is taken into the dynamical equation to compute the time evolution. The choice of evolution equations, such as the master equation and Schrödinger equation, depends on the specific physical problem and optimization task. Here we consider a closed quantum system and use the PWC approach for the time evolution. Note that this could be upgraded to other finite-difference methods to obtain the more

precise solution of the Schrödinger equation. Here the evolution operator at time t_k reads

$$U_k = e^{-iH[t_k]\Delta t} = e^{-i(H_d + \sum_j \tilde{\Omega}_j[t_k]H_j[t_k])\Delta t}. \quad (15)$$

Then the final evolution operator at time $t = T$ reads

$$U_T = U_N \dots U_1 U_0. \quad (16)$$

The final state S_T reads

$$S_T = U_T S_0. \quad (17)$$

C. Pulse distortion in the numerical process

There are at least three steps of pulse distortion in a complete quantum control task. First, the pulse is distorted from the waveform optimized numerically because of the finite sampling rate of the arbitrary waveform generator (AWG) [60,61]. Second, the pulse experiences distortion during the transmission due to impedance mismatching and other realistic filtering effects [27,28,57,62]. Third, to force the optimized pulses to satisfy physical constraints, pulse distortion is often induced when postprocessing the output pulses from optimization iteration, such as adding filter functions to the output. As mentioned above, PWC algorithms, such as GRAPE and Krotov, generates rough pulses. To smoothen the optimal pulses, low-pass filters such as a Gaussian filter [63] could be applied to suppress or cut off the high-frequency components and limit the bandwidth, after which the resultant waveform deforms and the fidelity is lowered from the optimum. On the other hand, limiting pulse amplitude [14] by applying a constraint function to the optimization results in distortion from the analytical form and induces additional high-frequency components. However, COCOA introduces all the pulse constraints into the optimization before the DFT node and all the high-frequency components will be filtered, overcoming this kind of pulse distortion perfectly. Additional customized constraints could be incorporated as well. This is enabled with the use of autodifferentiation in TensorFlow, which is discussed next. As a result, the optimized pulse is band-limited, maximum-limited, starting and ending at ZERO, while a definite analytical waveform is guaranteed to output from the algorithm. This is illustrated in Sec. IV

D. Autodifferentiation

The autodifferentiation (AD) method is widely used in machine learning, which is almost as accurate as symbolic differentiation [64]. There are two points of necessity that we choose AD: (1) AD obtains the derivatives to all inputs in one backpropagation when AD works in the reverse mode. So it is much more efficient than manual and symbolic differentiation and is more precise than numerical differentiation. (2) The complexity of derivatives induced by the bandwidth control and the extraction of computational subspace places many difficulties, such as expression swell problems, to manual and symbolic differentiation. The feasibility of autodifferentiation is demonstrated by the fact that each node of COCOA is derivable theoretically, so the total transform function of inputs to cost function is $\mathbb{F} : \mathbb{R}^n \rightarrow \mathbb{R}$. This is suitable for

the reverse mode of AD because the dimension of inputs is larger than outputs. We explicitly elaborate the process of autodifferentiation in COCOA optimization in Fig. 2. There are two processes when using autodifferentiation in reverse mode. (i) Forward propagation: when computing from $\{\Omega_j[k]\}$, $k = 1, \dots, N$ to F , it automatically constructs a computational graph formed of nodes and edges, as shown in Fig. 2 (solid line). (ii) Backpropagation: The gradients of F versus all inputs $\{\Omega_j[k]\}$, $k = 1, \dots, N$ are calculated with the computational graph by using a chain rule, as shown in Fig. 2 (dashed line). We note that all the derivatives of F with respect to inputs $\{\Omega_j[k]\}$, $k = 1, \dots, N$ are calculated in one backpropagation, which makes AD more efficient than other methods. Our numerical simulation results in Sec. IV will demonstrate the efficiency of AD in quantum optimal control. For more details of AD, please refer to [64].

IV. APPLICATIONS ON SUPERCONDUCTING QUBITS

To demonstrate the advantage of COCOA, we apply this algorithm to tackle one of the most challenging control obstacles in the up-to-date multiqubit processors: the always-on couplings. Such issue lies in many qubit systems such as superconducting qubits [12,65,66], quantum dots [67–69], NMR qubits [70,71], etc. Scaling-up qubit systems tends to reduce the number of control degrees, which means taking out more control fields out of the system. Losing either the control of qubit frequencies or coupling strength brings more difficulty in realizing good quantum gates, especially in the systems such as fixed frequency qubits with fixed couplings, tunable qubits with residual couplings, and qubits with tunable couplings with unwanted interaction and crosstalk [72–74]. Fortunately, the degree of control freedom on the pulse shaping could be further exploited with the help of COCOA. In this section, without loss of generality, we consider two realistic models of multiconnected superconducting qubits and applying COCOA to find optimal control pulses for single-qubit and two-qubit gates for the always-coupled qubits.

Before introducing the numerical simulations, we need to note two important tricks on how to choose the gradient optimization method when using COCOA for pulse optimization: 1. The Adam algorithm is always preferred when doing a global optimization, since Adam is more suitable for a broader search range due to its momentum factor and adaptive learning rate when updating variables [64]. We use the Adam algorithm for most numerical simulations except the CNOT pulse. 2. To obtain fast convergence in a local optimization scenario, see Fig. 6(b), choosing the SGD algorithm (stochastic gradient descent algorithm) with a small learning rate is favored, which is taken to be $\alpha = 0.001$. This is what we do when optimizing the CNOT pulse in Sec. IV B 1. For more details about these gradient algorithms, please refer to Ref. [75].

A. Model 1: Two transmon qubits coupled directly

In X-mon (X-shaped transmon) arrays the qubits are coupled directly via a capacitance with a constant interaction g , such as Google's previous version of the quantum computing chip Bristlecone and other chip designs with few control lines [13]. As a simplified model, we consider two qubits coupled

directly and obtain the Hamiltonian as

$$H_0 = \sum_{j=1,2} \omega_j a_j^\dagger a_j + \frac{\alpha_j}{2} a_j^\dagger a_j^\dagger a_j a_j + g_{12} (a_1^\dagger + a_1)(a_2^\dagger + a_2). \quad (18)$$

This model is effectively valid for qubits coupled via tunable couplers [12,76]. Here ω_j are the qubit frequencies and α_j are the anharmonicities of transmon qubits, $j = 1, 2$. To implement single-qubit operations, neighbor qubits are detuned with $\Delta = \omega_1 - \omega_2$ and the effective zz -coupling strength is turned down with the rate $\frac{g_{12}^2}{\Delta}$ [12,77–79]. g_{12} is the capacitive coupling strength between two qubits. $a^\dagger(a)$ denotes the qubit creation (annihilation) operators. This unwanted coupling gives rise to frequency splitting between $|00\rangle \rightarrow |01\rangle$ and $|10\rangle \rightarrow |11\rangle$, inducing gate errors, as well as control crosstalk [12]. This could be a more challenging issue when qubit frequencies are fixed [13]. Complex control pulses are proposed to resolved this issue, but finding appropriate pulses remains a difficulty [12,15]. The microwave pulses are sent in to drive the transmons via this operator:

$$H_d^j = a_j^\dagger e^{-i\omega_d t} + a_j e^{i\omega_d t}, \quad (19)$$

where ω_d is the driving frequency. The waveform $\Omega(t)$ is applied to the drive and modulates the strength of the control pulse. Hence, the total Hamiltonian reads

$$H = H_0 + \sum_j \Omega_j(t) H_d^j. \quad (20)$$

The control field could be added to both qubits 1 and 2 simultaneously or only on a single qubit 1 or 2.

To illustrate the properties of COCOA's solutions and demonstrate the advantage of the algorithm, we show some numerical examples of optimizing quantum gates in a realistic system by comparing different algorithms, including COCOA, CRAB, and GRAPE. For a fair comparison, we use the gradient descent optimizer Adam [75] in all these algorithms but keep the rest of the steps the same as the original versions. Therefore we denote them as COCOA, GRAPE-like, and CRAB-like in our results. In the simulation, each transmon is truncated to a four-level system to better consider leakage. The model parameter we used is similar to Ref. [80] as $\omega_1/2\pi = 5.270$ GHz, $\omega_2/2\pi = 4.670$ GHz, $\alpha_1/2\pi = \alpha_2/2\pi = 220$ MHz, $g_{12}/2\pi = 25.4$ MHz. The initial pulses for all algorithms are identical and take the form as shown in Eq. (14).

Note that the coupling strength between the two qubits is in the order of $g/\omega \simeq 10^{-2}$. The results for the weak coupling $g/\omega \leq 10^{-3}$ and ultrastrong coupling regime $g/\omega \geq 10^{-1}$ are shown in Appendix B, all of which demonstrates the enhancement of COCOA in the search of optimal pulses.

1. Single-qubit X gate at presence of interaction

The first gate is a single X rotation only on the second qubit while remaining the state in the first qubit. The target evolution operator of the two-qubit system is

$$U_{tar} = I \otimes \sigma_x, \quad (21)$$

which includes both qubit's dynamics in the computational subspace $\text{span}\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. The identity I in the

first qubit's subsystem meets the requirement that the control field does net-zero operation to the first qubit at the presence of crosstalk due to the coupling. Applying simple pulses results in entanglement between the two qubits. However, COCOA can efficiently find composite pulses to achieve the U_{tar} evolution. Here we set ω_d in resonance with transition $|00\rangle \rightarrow |01\rangle$; then it is off-resonant with $|10\rangle \rightarrow |11\rangle$ [12]. Other relevant parameters are $T = 50$ ns, $N = 148$, $f_s/2\pi = N/T = 2.96$ GHz, $N_c = 5$. The range of pulse amplitudes is $[-30, 30]$ MHz $\times 2\pi$.

We can conclude three advantages of COCOA according to the comparison result in Fig. 3: (1) Accuracy. As shown in Fig. 3(c), COCOA achieves the gate infidelity below 10^{-4} , which is the same order of magnitude with GRAPE-like's result and better than CRAB-like's. Even better results could be obtained by enlarging N_c , as illustrated later in Sec. IV A 3. (2) Smoothness and bandwidth. As shown in Fig. 3(a), the optimal pulse obtained by COCOA shows the best smoothness and limited pulse amplitude within $[-20, 20]$ MHz $\times 2\pi$. From the frequency spectrum of optimized pulse, as shown in Fig. 3(b), COCOA has limited bandwidth but the other's consist of high-frequency components. Interestingly, GRAPE-like's pulse shows a similar profile as COCOA, with more high-frequency components. Although CRAB-like's pulse looks smooth too, its amplitude is significantly stronger than the other two. (3) Analyticity. As promised, COCOA gives a definite analytical summation form of the basis, and all the high-frequency components are filtered completely. Specifically, COCOA outputs the pulse parameters for Eq. (14) with

$$a_0 = 5.066,$$

$$A_1, \dots, A_5 = -11.66, -4.172, -5.753, 2.140, 3.497,$$

$$\phi_1, \dots, \phi_5 = 1.080, -3.385, 6.104, -1.458, 1.098.$$

GRAPE-like and CRAB-like pulses both produce uncontrollable high-frequency components and cannot obtain definite analytical expressions due to the ZERO starting and ending point constraints.

COCOA shows a great advantage here, because the smoothness of the control pulse is rather important in many qubit systems where leakage and crosstalk errors are significant. Limiting the bandwidth could also help reduce pulse distortion throughout all the control steps. Moreover, the analytical expression with definite summation of a few chopped basis brings convenience and simplicity to the experimental adjustments. It is necessary to point out that, in principle, CRAB also uses an analytical waveform with definite summation of the chopped random basis, but the pulse constraints during the optimization process induces numerical pulse distortion from the original analytical form, leading to higher-frequency components, as shown in Fig. 3(b).

2. Dual X gate at the presence of interaction

Simultaneously driving coupled qubits is a challenging task, even in a tunable qubit system where the interaction could be turned on and off approximately. Crosstalk of control signals and ZZ interactions causes a significant drop in gate

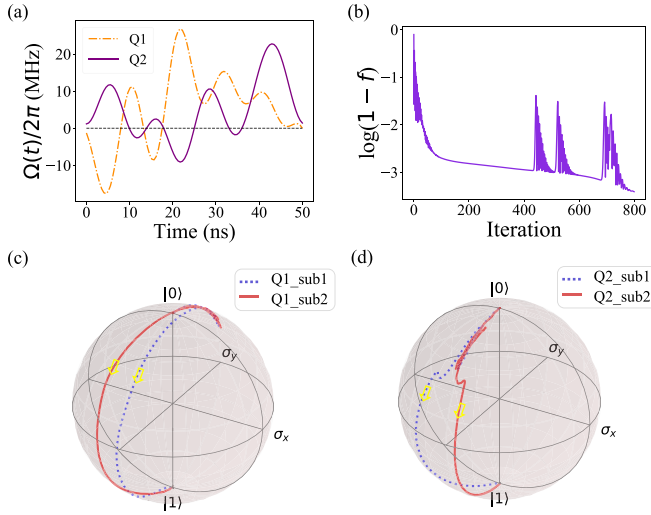


FIG. 4. Dual X gate optimization using COCOA. (a) Optimized control pulse for each qubit with pulse amplitude limitation $[-30, 30]$ MHz and ZERO starting and ending point. (b) Gate infidelity changes vs iteration, indicating the efficiency of the algorithm. (c) Evolution trajectories of qubit 1, driven by the dark orange (dash-dotted line) pulse in (b). Blue dotted (red solid) line shows the evolution in on-resonant (off-resonant) subspace span $\{|00\rangle, |01\rangle\}$ ($\{|10\rangle, |11\rangle\}$). (d) Evolution trajectories of qubit 2, driven by the purple (solid line) pulse in (b). Blue and red lines have the same meaning as in (c)

error compared to individual driving cases. To implement a high-fidelity dual (simultaneous) X gate on both nearest-coupled coupled qubits, we apply COCOA to find the pulses to drive them at the same time. The dual X gate simultaneously flips the states of the first and second qubits. The target evolution operator of the two-qubit system is

$$U_{tar} = \sigma_x \otimes \sigma_x. \quad (22)$$

Both qubits should be driven simultaneously and the drive term follows the same form of Eq. (19). Hence the total Hamiltonian reads

$$H = H_0 + \Omega_1(t)H_d^1 + \Omega_2(t)H_d^2. \quad (23)$$

Here we use the COCOA algorithm to find the optimized driving pulse of the dual X gate and we set $T = 50$ ns, $N = 200$, $N_c = 5$, $f_s = N/T = 4$ GHz. As we can see in Fig. 4, the optimized gate fidelity is greater than 99.9%, and the pulse parameter for each drive is given as follows:

Pulse parameters for Q1:

$$\begin{aligned} a_0 &= 5.091 \\ A_1, \dots, A_5 &= -10.46, 2.971, 3.033, 6.725, -6.124, \\ \phi_1, \dots, \phi_5 &= 5.858, 1.547, 3.085, 1.458, -3.650 \end{aligned}$$

Pulse parameter for Q2:

$$\begin{aligned} a_0 &= 5.078 \\ A_1, \dots, A_5 &= 7.790, 0.8855, 4.857, -6.030, 1.787 \\ \phi_1, \dots, \phi_5 &= 0.7156, 1.714, 2.507, 0.8538, 2.292 \end{aligned}$$

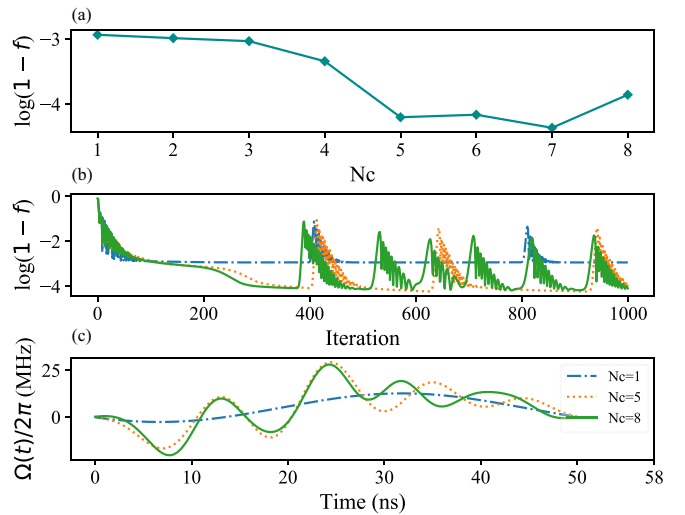


FIG. 5. Explore how N_c affects the accuracy, efficiency, and smoothness of our algorithm. (a) Best infidelity that COCOA can reach with each N_c . (b) Infidelity changes vs iterations for different N_c , indicating the efficiency. (c) The complexity of optimized pulse change vs N_c .

By analyzing the driving pulse and its corresponding Bloch trajectory, we found that the negative part of the driving pulse eliminates the detuning of the off-resonance subspace with respect to on-resonance subspace, namely, the ZZ coupling.

3. Optimizing N_c

The key parameter N_c , i.e., the number of Fourier components, determines the smoothness and bandwidth of the pulse, as well as the number of optimized parameters, which increases at a scaling rate of $2N_c$. Consequently, the choice of N_c affects the optimization efficiency and accuracy.

We take the previous case in Sec. IV A 1 as an example to study how N_c affects the optimization, where we only tune N_c while fixing all the other parameters. As shown in Fig. 5(a), gate infidelity is improved by one order of magnitude when N_c increases from 1 to 5 and does not improve much beyond 5. Figures 5(b) and 5(c) demonstrate the convergence behavior and pulse shape of $N_c = 1, 5, 8$. From these simulation results, we observe that $N_c = 5$ is the best choice based on the consideration of the tradeoff between the number of parameters and optimization accuracy.

Theoretically, if there is no bandwidth limit, the COCOA algorithm can approach to GRAPE algorithm when N_c reaches its maximum: $N_c^{max} = \text{Int}(\frac{N-1}{2})$, where $\text{Int}(\cdot)$ means rounding down.

B. Model 2: Two transmon qubits coupled via a cavity

In another widely used architecture, superconducting qubits are coupled via superconducting cavities, such as one-dimensional transmission line resonators [81–83], with the

Hamiltonian

$$H_0 = \sum_{j=1,2} \omega_j a_j^\dagger a_j + \frac{\alpha_j}{2} a_j^\dagger a_j^\dagger a_j a_j + g_{cj} (b^\dagger a_j + b a_j^\dagger) + \omega_c b^\dagger b, \quad (24)$$

where ω_c is the frequency of the cavity. g_{cj} is coupling strength between the resonator and the j th qubit. $b(b^\dagger)$ is the annihilation (creation) operator of the resonator. Other parameters have the same meaning as in model 1. We take $\omega_1/2\pi = 6.2$ GHz, $\omega_2/2\pi = 6.8$ GHz, $\omega_c/2\pi = 7.15$ GHz, $\alpha_{1,2}/2\pi = 350$ MHz, $g_{1,2}/2\pi = 250$ MHz, which are used in Ref. [9]. The drive Hamiltonian H_d has the same form as in Eq. (19).

Since the cavity behaves as merely a larger scale fixed-coupler between two qubits, control pulses for single-qubit gates could be obtained similarly as in a previous discussion. Detailed numerical results of a single-qubit X gate could be found in Appendix C. Here we demonstrate the optimization of a two-qubit entangling gate for this model.

1. Optimizing CNOT gate based on the SWIPHT protocol

It is worth pointing out that COCOA can fully utilize the prior knowledge of analytical methods and obtain completely analytical optimal pulses via local optimization around an analytically given pulse. To demonstrate this, we start from a CNOT gate implementation using the SWIPHT protocol (speeding up wave forms by inducing phases to harmful transitions) [9,10,13]. The given analytical form of the pulse is

$$\Omega(t) = \frac{\ddot{\chi}}{2\sqrt{\frac{\delta^2}{4} - \dot{\chi}^2}} - \sqrt{\frac{\delta^2}{4} - \dot{\chi}^2} \cot(2\chi), \quad (25)$$

where $\chi(t) = \frac{A}{T^8} t^4 (T-t)^4 + \frac{\pi}{4}$, $A = 138.9$, $T = 5.87/|\delta|$. $\delta = \omega_{\tilde{1}0 \rightarrow \tilde{1}1} - \omega_{\tilde{0}0 \rightarrow \tilde{0}1}$ is the detuning between the target and the harmful transition in the computational subspace $\text{span}\{|\tilde{0}1\rangle, |\tilde{0}\bar{1}\rangle, |\tilde{1}0\rangle, |\tilde{1}\bar{1}\rangle\}$, where the first qubit is the control qubit and the second one is the target qubit. The CNOT operator generated with a single microwave control could be expressed as this general form,

$$U_{\text{tar}} = (\sigma_x \oplus I) \prod_{i=1,2; j=x,y,z} R_{ij}(\theta_{ij}), \quad (26)$$

where $R_{ij}(\theta_{ij})$ are single-qubit rotations with arbitrary angles for optimization. This U_{tar} is equivalent to a standard CNOT $\sigma_x \oplus I$ up to some local phases. We set the drive frequency $\omega_d = \omega_{\tilde{0}0 \rightarrow \tilde{0}1}$, and $N_c = 9$. Figure 6(b) shows that the local optimization converges very fast with prior knowledge of the optimal pulse, which demonstrates the advantage of numerical optimization based on analytically optimal pulses. Figure 6(a) shows that the optimized CNOT pulse is transformed but still maintains a similar shape as the initial analytically optimal pulse. We note that the optimized pulse, shown in Fig. 6(a), can be generated more accurately with an AWG device due to its limited bandwidth that the initial pulse does not process. With the same evolution time $T = 35.4$ ns, we finally obtained the optimized driving pulse shown in Fig. 6(a) with complete

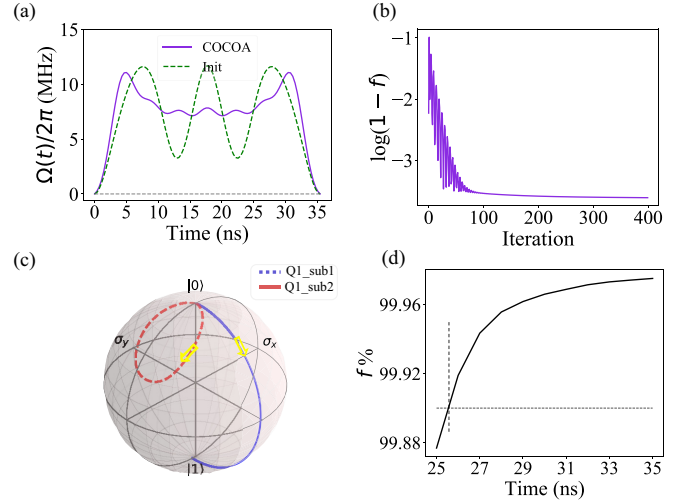


FIG. 6. CNOT gate optimization using COCOA. (a) Pulse comparison between the initial SWIFT CNOT pulse (green-dashed line) in Ref. [9] and our optimized CNOT pulse (purple-solid line). (b) Gate infidelity changes vs iteration, indicating the efficiency of the algorithm. (c) Evolution trajectory on Bloch sphere driven by our optimized pulse in (b). The blue and red lines show the target qubit's evolution trajectory in the two subspaces $\text{span}\{|00\rangle, |01\rangle\}$ and $\text{span}\{|10\rangle, |11\rangle\}$. (d) The gate speed limit of CNOT.

pulse parameters (in units of MHz) in Eq. (14):

$$\begin{aligned} a_0 &= 7.416, \\ A_1, \dots, A_9 &= -0.818, -2.05, -2.27, -1.50 - 0.807, \\ &\quad -0.2020.0287, 0.325, 0.00291, \\ \phi_1, \dots, \phi_9 &= 0, \pi, \pi, 0, 0, 0, 0.000138, 0, 0.00198. \end{aligned}$$

The speed of this analytical CNOT gate is limited by $\frac{\delta^2}{4} - \dot{\chi}^2 > 0$ derived from Eq. (25). Hence, $T > 0.02975 \frac{A}{\delta}$, and we have $T_{\text{min}} = 24.95$ ns when $A = 138$ and $\delta = 26.4$ MHz $\times 2\pi$. The behavior of the optimal CNOT gate approaching the speed limit is shown in Fig. 6(c). Here we start from $T = 25$ ns and increase by 1 ns each step to observe the change of the optimal fidelity with the gate time. Finally, we see that the gate fidelity exceeds 99.9% when $T \geq 26$ ns, which is a significant improvement from the initial CNOT gate time of 35 ns using SWIFT theory.

V. CONCLUSION

In this paper we have developed an algorithm to optimize smooth quantum control pulse constraints, which could be very complex, highly nonlinear, sub-/superdifferentiable approximations, and optional. In the particular demonstration examples, we limit the pulse amplitude, pulse bandwidth, and the number of pulse parameters. Doing so makes this algorithm involve complicated computations for the differentiation of the expectation function versus the optimizing parameters. We resolve this issue using autodifferentiation powered by TensorFlow. Therefore this algorithm can be straightforwardly extended to larger quantum systems with even more complicated calculations of gradients. We have demonstrated the advantages of the proposed algorithm by applying it to

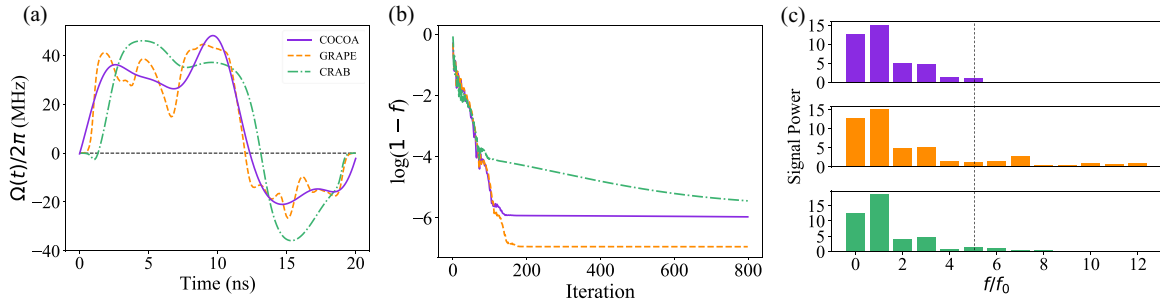


FIG. 7. Algorithm comparison. $g = 1 \text{ MHz} \times 2\pi$. (a) Optimized pulse with zero amplitude at $t = 0 \text{ ns}$ and $t = 20 \text{ ns}$. Pulse amplitude is limited between $[-40, 40] \text{ MHz}$. (b) Gate infidelity changing vs iteration, showing the efficiency of each algorithm. (c) Frequency spectrum of each pulse. $f_0 = 1/T = 0.05 \text{ GHz}$. As we can see, the GRAPE [orange dashed line in (a)] and CRAB [green dash-dotted line in (a)] have more high-frequency components induced by pulse constraints. The gray dashed line shows the truncated components of the COCOA algorithm.

realistic superconducting qubit models with always-on ZZ interaction and achieving optimal smooth pulses to implement single-qubit gates and two-qubit gates. Compared to the GRAPE and CRAB algorithms, we obtain higher gate fidelities and better optimization efficiency. We have shown that COCOA could be applied to the optimization scenarios either with or without good prior knowledge by simply switching the optimizers and obtaining high-fidelity gates for both cases.

We summarize COCOA's advantages as follows: 1. COCOA outputs optimal pulses with definite analytical expression. 2. The optimal pulses have manually limited bandwidth and amplitude. 3. This algorithm is more compatible with complex and flexible pulse constraints, without the induced pulse distortion in numerical optimization. 4. The autodifferentiation assisted by TensorFlow enables efficient and easy calculation of gradients and the ability to handle complex computing processes and complex models. 5. COCOA can speed up until the optimization by locally searching the optimal pulses based on certain prior knowledge. Using COCOA, we completely resolve the challenging problem, to implement either individually or simultaneously a single-qubit X gate in a strongly ZZ-interacting two-qubit system [12,13,73]. In conclusion, COCOA optimization is friendly to realistic quantum control tasks, easily customizable, and easily extended to larger quantum systems.

In conclusion, COCOA provides a versatile, highly functional, and efficient platform to add physical constraints into

quantum optimal control tasks. Following the line of COCOA, more work could be pursued in the near future to resolve the realistic QOC issues. For example, pulse predistortion could be effectively performed by adding the definite transfer function of control lines and pulse generators into the optimization process. Different analytical forms with fewer pulse parameters could be investigated using the proposed algorithm so that the numerical approach could better meet experimental needs.

ACKNOWLEDGMENTS

This work was supported by the Key-Area Research and Development Program of Guang-Dong Province (Grant No. 2018B030326001), the National Natural Science Foundation of China (U1801661), the Guangdong Innovative and Entrepreneurial Research Team Program (2016ZT06D348), the Guangdong Provincial Key Laboratory (Grant No. 2019B121203002), the Natural Science Foundation of Guangdong Province (2017B030308003), the Science, Technology, and Innovation Commission of Shenzhen Municipality (JCYJ20170412152620376, KYTDPT20181011104202253), and the NSF of Beijing (Grant No. Z190012), Shenzhen Science and Technology Program (KQTD20200820113010023).

Y.S. and X.-H.D. conceived the project, designed the algorithm, performed calculations, and wrote the manuscript. Y.S. did all the coding. J.L. gave some important suggestions on the algorithm and helped design the flowchart. Y.-J.H. helped

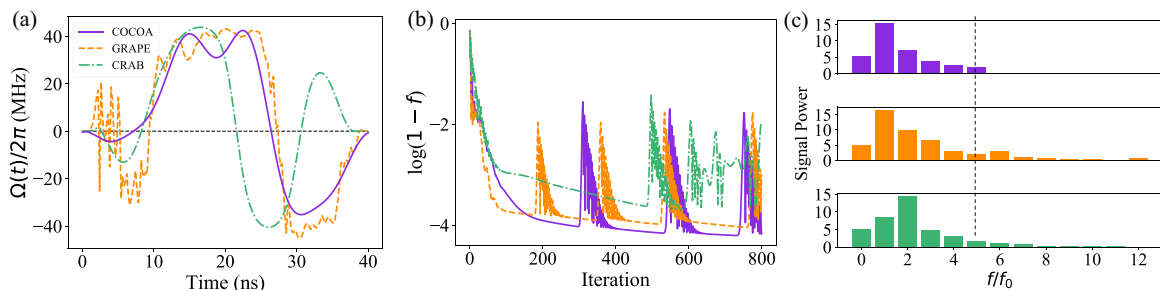


FIG. 8. Algorithm comparison. $g = 100 \text{ MHz} \times 2\pi$. (a) Optimized pulse with zero amplitude at $t = 0 \text{ ns}$ and $t = 20 \text{ ns}$. The pulse amplitude is limited between $[-40, 40] \text{ MHz}$. (b) Gate infidelity changing vs iteration, showing the efficiency of each algorithm. (c) Frequency spectrum of each pulse. $f_0 = 1/T = 0.025 \text{ GHz}$. As we can see, GRAPE [orange dashed line in (a)] and CRAB [green dash-dotted line in (a)] have more high-frequency components induced by pulse constraints. The dashed line shows the truncated components of the COCOA algorithm.

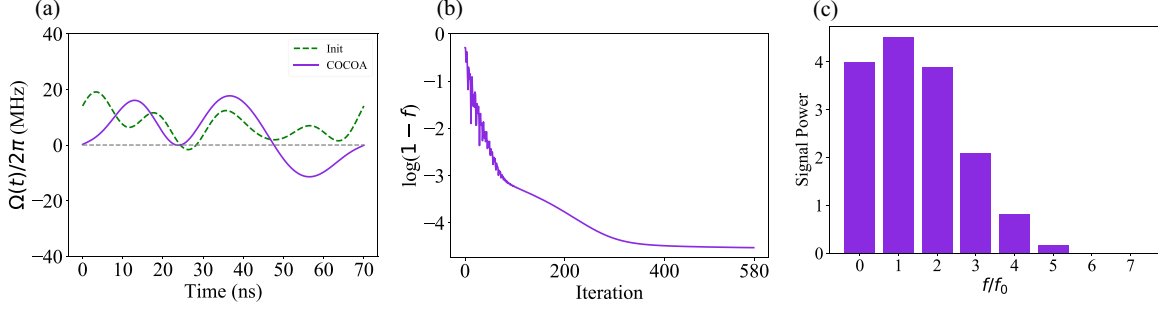


FIG. 9. Single X gate for model 2. (a) Optimized pulse with zero amplitude at $t = 0$ ns and $t = 70$ ns. Pulse amplitude is limited between $[-20, 20]$ (unit: MHz). The purple solid line is the optimized pulse with the COCOA algorithm, and the green dashed line is the initial Fourier pulse. (b) Gate infidelity changing vs iteration, showing the efficiency of the algorithm. (c) Frequency spectrum of COCOA pulse with limited bandwidth ($N_c = 5$). $f_0 = \frac{1}{T} = 0.0143$ GHz.

with refining the algorithm. Q.G. assisted with the GRAPE optimization. X.-H.D. oversaw the project.

APPENDIX A: LOW-PASS FILTERING

DFT. Given a discretized pulse sequence in time domain $\Omega_j[k]$, $k = 1 \dots N$. Note that the pulse here can be any form or without any form. After DFT we can get a complex sequence that contains amplitude and phase information:

$$X[n] = \sum_{k=1}^N \Omega_j[k] e^{i(2\pi n/N)k}. \quad (\text{A1})$$

We can deduce that

$$X[N-n] = \sum_{k=1}^N \Omega_j[k] e^{-j(2\pi/N)(N-n)k} \quad (\text{A2})$$

$$= \sum_{k=1}^N \Omega_j[k] e^{j(-2k\pi + (2\pi/N)kn)} \quad (\text{A3})$$

$$= \sum_{k=1}^N \Omega_j[k] e^{j(2\pi/N)kn} \quad (\text{A4})$$

$$= X^*[n]. \quad (\text{A5})$$

This is an important characteristic of the complex sequence.

iDFT:

$$\tilde{\Omega}_j[k] = \frac{1}{N} \sum_{n=1}^N Y[n] e^{i(2\pi k/N)n}. \quad (\text{A6})$$

Without loss of generality, we assume N is an even number, and then we obtain the pulse amplitude in the time domain as

$$N \times \Omega_j[k] = \sum_{n=1}^N X[n] e^{j(2\pi/N)kn} \quad (\text{A7})$$

$$= X[0] + X[1]e^{j(2\pi/N)} + X[2]e^{j(2\pi/N)2} + \dots$$

$$+ X[N/2-1]e^{j\left(\frac{2\pi}{N}(N/2-1)\right)} + X[N/2]$$

$$+ X[N/2+1]e^{j\left(\frac{2\pi}{N}(N/2+1)\right)} + \dots$$

$$+ X[N-2]e^{j\left(\frac{2\pi}{N}(N-2)\right)} + X[N-1]e^{j\left(\frac{2\pi}{N}(N-1)\right)}. \quad (\text{A8})$$

We set $X[n] = a_n + ib_n$. Combining the sum of the conjugate symmetric terms $X[n]$ and $X[N-n]$, we obtain

$$(a_n + ib_n)e^{j(2\pi/N)kn} + (a_n - ib_n)e^{j(2\pi/N)(N-k)n}$$

$$= A_n \cos\left(\frac{2kn\pi}{N} + \phi_n\right), \quad (\text{A9})$$

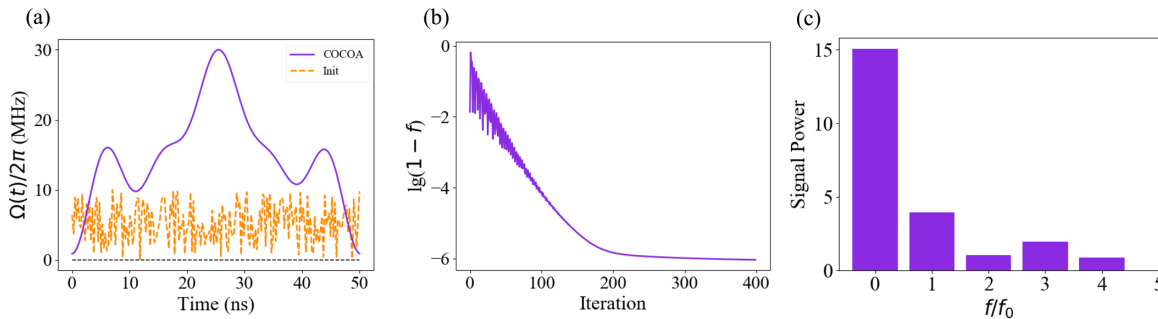


FIG. 10. Efficiency of COCOA algorithm with a random initial pulse sequence. This numerical simulation is similar to Sec. IV A 1, except for the initial pulse. (a) Optimized pulse with zero amplitude at $t = 0$ ns and $t = 50$ ns. Pulse amplitude is limited between $[-30, 0]$ MHz. Dashed line is the random initial pulse sequence, and each element is in the range of $[0, \pi/T]$. (b) Gate infidelity changing vs iteration, showing the efficiency of COCOA, even starting with a random initial pulse. (c) Frequency spectrum of COCOA pulse with cut-off frequency $4f_0$, where $f_0 = 1/T = 0.02$ GHz.

where $A_n = 2\sqrt{a_n^2 + b_n^2}$ and $\tan(\phi_n) = \frac{b_n}{a_n}$. Then Eq. (A8) becomes

$$\begin{aligned} N \times \Omega_j[k] = & a_0 + A_1 \cos(n2\pi/N + \phi_1) \\ & + A_2 \cos[(2n)2\pi/N + \phi_2] + \dots \\ & + A_{\frac{N}{2}-1} \cos([(N/2 - 1)n]2\pi/N + \phi_{\frac{N}{2}-1}). \end{aligned} \quad (\text{A10})$$

Assuming the sample frequency is f_0 , we have

$$\Omega_j[k] = \Omega_j[kT_0] = \Omega_j(k/f_0). \quad (\text{A11})$$

Replacing k with $f_0 t$, we then have

$$\begin{aligned} N \times \Omega_j(t) = & a_0 + A_1 \cos(f_0 2\pi t/N + \phi_1) \\ & + A_2 \cos[(2f_0)2\pi t/N + \phi_2] + \dots \\ & + A_{\frac{N}{2}-1} \cos([(N/2 - 1)f_0]2\pi t/N + \phi_{\frac{N}{2}-1}). \end{aligned} \quad (\text{A12})$$

APPENDIX B: WEAK AND STRONG COUPLING STRENGTH

This Appendix demonstrates that the COCOA algorithm can still be efficient in different coupling strengths. In Fig. 7

we show the algorithm comparison with $g = 0.001 \text{ GHz} \times 2\pi$. Figure 8 shows the result of $g = 0.1 \text{ GHz} \times 2\pi$.

APPENDIX C: SINGLE-QUBIT X GATE WITH ALWAYS-ON INTERACTION FOR MODEL 2

Here we show the result of a single-qubit X gate in model 2 using the COCOA algorithm to show its power and compatibility with a multiqubit system. The result is shown in Fig. 9.

APPENDIX D: INITIAL PULSES AS A RANDOM PWC SEQUENCE

As mentioned in Sec. III B 1, the initial pulse in the COCOA algorithm can be of any kind, including a random sequence. Here we conduct a numerical simulation starting from a random PWC sequence based on model 1 with $g = 1 \text{ MHz} \times 2\pi$. The initial random sequence with each value ranging within $[0, 10]$ is generated using the random number generator in NUMPY. Compared to the convergence trend in Figs. 3(b) and 6(b), where the initial pulses are the random Fourier basis and SWIPHT pulse with prior knowledge, this simulation in Fig. 10 demonstrates that COCOA can always be efficient no matter what the initial pulse looks like.

-
- [1] A. Steane, *Rep. Prog. Phys.* **61**, 117 (1998).
- [2] C. L. Degen, F. Reinhard, and P. Cappellaro, *Rev. Mod. Phys.* **89**, 035002 (2017).
- [3] V. Giovannetti, S. Lloyd, and L. Maccone, *Nat. Photonics* **5**, 222 (2011).
- [4] T. Naranjo, K. M. Lemishko, S. de Lorenzo, Á. Somoza, F. Ritort, E. M. Pérez, and B. Ibarra, *Nat. Commun.* **9**, 4512 (2018).
- [5] P. Rembold, N. Oshnik, M. M. Müller, S. Montangero, T. Calarco, and E. Neu, *AVS Quantum Sci.* **2**, 024701 (2020).
- [6] Z. Chen, K. J. Satzinger, J. Atalaya, A. N. Korotkov, A. Dunsworth, D. Sank, C. Quintana, M. McEwen, R. Barends, P. V. Klimov *et al.*, *Nature* **595**, 383 (2021).
- [7] D. Stefanatos and E. Paspalakis, *Europhys. Lett.* **132**, 60001 (2020).
- [8] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm, *Phys. Rev. Lett.* **103**, 110501 (2009).
- [9] S. E. Economou and E. Barnes, *Phys. Rev. B* **91**, 161405 (2015).
- [10] X.-H. Deng, E. Barnes, and S. E. Economou, *Phys. Rev. B* **96**, 035441 (2017).
- [11] P. Doria, T. Calarco, and S. Montangero, *Phys. Rev. Lett.* **106**, 190501 (2011).
- [12] X.-H. Deng, Y.-J. Hai, J.-N. Li, and Y. Song, [arXiv:2103.08169](https://arxiv.org/abs/2103.08169).
- [13] J. Long, T. Zhao, M. Bal, R. Zhao, G. S. Barron, H.-s. Ku, J. A. Howard, X. Wu, C. R. H. McRae, X.-H. Deng *et al.*, [arXiv:2103.12305](https://arxiv.org/abs/2103.12305).
- [14] S. Machnes, E. Assémat, D. Tannor, and F. K. Wilhelm, *Phys. Rev. Lett.* **120**, 150401 (2018).
- [15] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, *J. Magn. Reson.* **172**, 296 (2005).
- [16] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquieres, A. Gruslys, S. Schirmer, and T. Schulte-Herbrüggen, *Phys. Rev. A* **84**, 022305 (2011).
- [17] A. Ruschhaupt, X. Chen, D. Alonso, and J. Muga, *New J. Phys.* **14**, 093040 (2012).
- [18] E. Zahedinejad, S. Schirmer, and B. C. Sanders, *Phys. Rev. A* **90**, 032310 (2014).
- [19] T. Caneva, T. Calarco, and S. Montangero, *Phys. Rev. A* **84**, 022326 (2011).
- [20] A. P. Peirce, M. A. Dahleh, and H. Rabitz, *Phys. Rev. A* **37**, 4950 (1988).
- [21] Y. Ohtsuki, G. Turinici, and H. Rabitz, *J. Chem. Phys.* **120**, 5509 (2004).
- [22] S. Lloyd and S. Montangero, *Phys. Rev. Lett.* **113**, 010502 (2014).
- [23] Y. Ohtsuki and K. Nakagami, *Phys. Rev. A* **77**, 033414 (2008).
- [24] K. Sundermann and R. de Vivie-Riedle, *J. Chem. Phys.* **110**, 1896 (1999).
- [25] D. Stefanatos and E. Paspalakis, *Phys. Rev. A* **102**, 013716 (2020).
- [26] M. Jerger, A. Kulikov, Z. Vasselin, and A. Fedorov, *Phys. Rev. Lett.* **123**, 150501 (2019).
- [27] I. N. Hincks, C. E. Granade, T. W. Borneman, and D. G. Cory, *Phys. Rev. Appl.* **4**, 024012 (2015).
- [28] M. A. Rol, L. Ciorciaro, F. K. Malinowski, B. M. Tarasinski, R. E. Sagastizabal, C. C. Bultink, Y. Salathe, N. Haandbæk, J. Sedivy, and L. DiCarlo, *Appl. Phys. Lett.* **116**, 054001 (2020).
- [29] M. Lapert, R. Tehini, G. Turinici, and D. Sugny, *Phys. Rev. A* **79**, 063411 (2009).
- [30] Z. Tošner, T. Vosegaard, C. Kehlet, N. Khaneja, S. J. Glaser, and N. C. Nielsen, *J. Magn. Reson.* **197**, 120 (2009).
- [31] J. A. Jones, *Prog. Nucl. Magn. Reson. Spectrosc.* **59**, 91 (2011).
- [32] C. Ryan, M. Laforest, and R. Laflamme, *New J. Phys.* **11**, 013034 (2009).
- [33] N. C. Nielsen, C. Kehlet, S. J. Glaser, and N. Khaneja, in *eMagRes* (John Wiley & Sons, 2007).

- [34] R. Schutjens, F. A. Dagga, D. J. Egger, and F. K. Wilhelm, *Phys. Rev. A* **88**, 052330 (2013).
- [35] A. Blais, S. M. Girvin, and W. D. Oliver, *Nat. Phys.* **16**, 247 (2020).
- [36] J. L. Allen, R. Kosut, J. Joo, P. Leek, and E. Ginossar, *Phys. Rev. A* **95**, 042325 (2017).
- [37] A. Blais, J. Gambetta, A. Wallraff, D. I. Schuster, S. M. Girvin, M. H. Devoret, and R. J. Schoelkopf, *Phys. Rev. A* **75**, 032329 (2007).
- [38] A. Blais, A. L. Grimsmo, S. Girvin, and A. Wallraff, *Rev. Mod. Phys.* **93**, 025005 (2021).
- [39] Z.-H. Wang and V. V. Dobrovitski, *Phys. Rev. B* **84**, 045303 (2011).
- [40] F. Platzer, F. Mintert, and A. Buchleitner, *Phys. Rev. Lett.* **105**, 020501 (2010).
- [41] S. van Frank, M. Bonneau, J. Schmiedmayer, S. Hild, C. Gross, M. Cheneau, I. Bloch, T. Pichler, A. Negretti, T. Calarco *et al.*, *Sci. Rep.* **6**, 34187 (2016).
- [42] M. Y. Niu, S. Boixo, V. N. Smelyanskiy, and H. Neven, *npj Quantum Inf.* **5**, 33 (2019).
- [43] C.-C. Shu, T.-S. Ho, X. Xing, and H. Rabitz, *Phys. Rev. A* **93**, 033417 (2016).
- [44] K. W. Moore and H. Rabitz, *J. Chem. Phys.* **137**, 134113 (2012).
- [45] R. Chakrabarti and H. Rabitz, *Int. Rev. Phys. Chem.* **26**, 671 (2007).
- [46] H. A. Rabitz, M. M. Hsieh, and C. M. Rosenthal, *Science* **303**, 1998 (2004).
- [47] A. N. Pechen and D. J. Tannor, *Phys. Rev. Lett.* **106**, 120402 (2011).
- [48] S. Kirchhoff, T. Keßler, P. J. Liebermann, E. Assémat, S. Machnes, F. Motzoi, and F. K. Wilhelm, *Phys. Rev. A* **97**, 042348 (2018).
- [49] M. Larocca and D. Wisniacki, *Phys. Rev. A* **103**, 023107 (2021).
- [50] J. Tian, H. Liu, Y. Liu, P. Yang, R. Betzholz, R. S. Said, F. Jelezko, and J. Cai, *Phys. Rev. A* **102**, 043707 (2020).
- [51] W. N. Plick and M. Krenn, *Phys. Rev. A* **92**, 063841 (2015).
- [52] J. Kelly, R. Barends, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, I.-C. Hoi, E. Jeffrey, A. Megrant, J. Mutus, C. Neill, P.J.J. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T.C. White, A.N. Cleland, and J. M. Martinis, *Phys. Rev. Lett.* **112**, 240504 (2014).
- [53] M. Rol, F. Battistel, F. Malinowski, C. Bultink, B. Tarasinski, R. Vollmer, N. Haider, N. Muthusubramanian, A. Bruno, B. Terhal *et al.*, *Phys. Rev. Lett.* **123**, 120502 (2019).
- [54] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, [arXiv:1603.04467](https://arxiv.org/abs/1603.04467).
- [55] L. H. Pedersen, N. M. Møller, and K. Mølmer, *Phys. Lett. A* **367**, 47 (2007).
- [56] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1975).
- [57] Z. Chen, *Metrology of Quantum Control and Measurement in Superconducting Qubits* (University of California, Santa Barbara, Santa Barbara, CA, 2018).
- [58] J. M. Martinis and M. R. Geller, *Phys. Rev. A* **90**, 022307 (2014).
- [59] D. d'Alessandro, *Introduction to Quantum Control and Dynamics* (CRC Press, Boca Raton, FL, 2007).
- [60] J. Lin, F.-T. Liang, Y. Xu, L.-H. Sun, C. Guo, S.-K. Liao, and C.-Z. Peng, [arXiv:1806.03660](https://arxiv.org/abs/1806.03660).
- [61] J. Raftery, A. Vrajitoarea, G. Zhang, Z. Leng, S. Srinivasan, and A. Houck, [arXiv:1703.00942](https://arxiv.org/abs/1703.00942).
- [62] Y. Baum, M. Amico, S. Howell, M. Hush, M. Liuzzi, P. Mundada, T. Merkh, A. R. Carvalho, and M. J. Biercuk, *PRX Quantum* **2**, 040324 (2021).
- [63] H. J. Blinichikoff and A. I. Zverev, *Filtering in the Time and Frequency Domains* (Krieger Publishing Co., Inc., 1986).
- [64] A. Güneş Baydin, B. A. Pearlmutter, A. Andreyevich Radul, and J. Mark Siskind, *J. Mach. Learn. Res.* **18**, 1 (2018).
- [65] P. Zhao, P. Xu, D. Lan, J. Chu, X. Tan, H. Yu, and Y. Yu, *Phys. Rev. Lett.* **125**, 200503 (2020).
- [66] J. Ku, X. Xu, M. Brink, D. C. McKay, J. B. Hertzberg, M. H. Ansari, and B. L. T. Plourde, *Phys. Rev. Lett.* **125**, 200504 (2020).
- [67] A. V. Khaetskii, D. Loss, and L. Glazman, *Phys. Rev. Lett.* **88**, 186802 (2002).
- [68] L. Jacak, P. Hawrylak, and A. Wojs, *Quantum Dots* (Springer Science & Business Media, New York, 2013).
- [69] P. A. Maksym and T. Chakraborty, *Phys. Rev. Lett.* **65**, 108 (1990).
- [70] L. M. Vandersypen and I. L. Chuang, *Rev. Mod. Phys.* **76**, 1037 (2005).
- [71] Y. Wang, A. Kumar, T.-Y. Wu, and D. S. Weiss, *Science* **352**, 1562 (2016).
- [72] M. Sarovar, T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, *Quantum* **4**, 321 (2020).
- [73] A. Ash-Saki, M. Alam, and S. Ghosh, *IEEE Trans. Quantum Eng.* **1**, 3101406 (2020).
- [74] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Association for Computing Machinery, New York, NY, 2020), pp. 1001–1016.
- [75] S. Ruder, [arXiv:1609.04747](https://arxiv.org/abs/1609.04747).
- [76] 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 (London)* **574**, 505 (2019).
- [77] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, *Appl. Phys. Rev.* **6**, 021318 (2019).
- [78] M. Kjaergaard, M. E. Schwartz, J. Braumüller, P. Krantz, J. I.-J. Wang, S. Gustavsson, and W. D. Oliver, *Annu. Rev. Condens. Matter Phys.* **11**, 369 (2020).
- [79] G. Wendin, *Rep. Prog. Phys.* **80**, 106001 (2017).
- [80] J. Qiu, Y. Zhou, C.-K. Hu, J. Yuan, L. Zhang, J. Chu, W. Huang, W. Liu, K. Luo, Z. Ni *et al.*, *Phys. Rev. Applied* **16**, 054047 (2021).
- [81] J. Majer, J. Chow, J. Gambetta, J. Koch, B. Johnson, J. Schreier, L. Frunzio, D. Schuster, A. A. Houck, A. Wallraff *et al.*, *Nature (London)* **449**, 443 (2007).
- [82] M. Gong, M.-C. Chen, Y. Zheng, S. Wang, C. Zha, H. Deng, Z. Yan, H. Rong, Y. Wu, S. Li, F. Chen, Y. Zhao, F. Liang, J. Lin, Y. Xu, C. Guo, L. Sun, A. D. Castellano, H. Wang, C. Peng, C. Y. Lu, X. Zhu, and J. W. Pan, *Phys. Rev. Lett.* **122**, 110501 (2019).
- [83] J. M. Chow, J. M. Gambetta, A. D. Corcoles, S. T. Merkel, J. A. Smolin, C. Rigetti, S. Poletto, G. A. Keefe, M. B. Rothwell, J. R. Rozen, M.B. Ketchen, and M. Steffen, *Phys. Rev. Lett.* **109**, 060501 (2012).