Black-Box Optimization and FPGA-based Feedback

Sara Sussman

(Dated: April 2019)

Black-box quantum gate optimization is a scalable tool for quantum optimal control because it does not require detailed knowledge of the system Hamiltonian. In this paper I will describe a variety of algorithms that could be used for high-dimensional black-box optimization and detail their experimental pros and cons. I will conclude with an overview of how FPGAs are used in quantum experiments.

Superconducting qubit systems optimized by quantum optimal control (QOC) algorithms have yielded interesting physics. For example, entangling gates have been optimized to allow for the realization of a full universal set of quantum gates with minimized errors and durations [1]. More recently, a QOC algorithm implemented an artificial neuron by employing an "update rule" that trained a machine learning algorithm [2]. Other interesting directions in QOC involve ultra-fast pulse shaping and optimizing qubit state preparation while accounting for dissipation and qubit leakage [3-6].

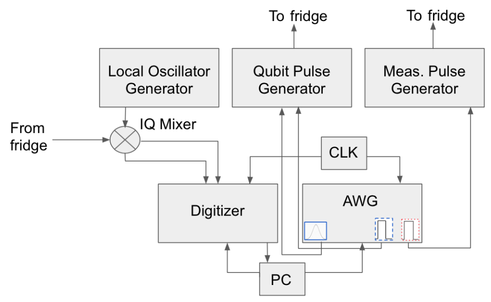

FIG. 1: Setup for a qubit experiment. The AWG outputs control pulses that specify the quantum gate. These pulses are upconverted to microwave frequencies by mixing with carriers via vector signal generators before they enter the fridge. The qubit pulse acts on the qubit, and the measurement pulse is modified by the resulting qubit state. The I/Q mixer demodulates the modified measurement pulse and the measurement I and Q waveforms are digitized and read into the PC. The qubit state is computed from the digitizer data in software. Depending on the qubit state, the next set of pulses are loaded into the AWG for the next experiment.

In circuit QED, a microwave qubit pulse acts as a quantum gate. Ultimately, this qubit pulse can be decomposed into signals from an arbitrary waveform generator (AWG). Fast AWG sampling rates allow for direct digital synthesis schemes where the AWG directly outputs the qubit pulse [7]. In more typical qubit experiment setups, the AWG outputs two Hann window functions I and Q (in-phase and quadrature) that modulate the fast baseband of the I and Q signals, respectively. These control signals are then upconverted to microwave frequencies (Figure 1). To optimize the qubit pulse, over j iterations we modulate the sets of amplitudes $\{A_m^j\}$, $\{B_m^j\}$ of the I/Q window functions as in Equation 1:

The j-th iteration of I and Q window functions for a n-dimensional loss function, where I and Q each have $n/2$ Fourier components:

$$\begin{aligned} I^j &= \textstyle\sum_{m=1}^{n/2} A_m^j\left(1-\cos\left(\frac{2\pi i t}{T_m}\right)\right) \\ Q^j &= \textstyle\sum_{m=1}^{n/2} B_m^j\left(1-\cos\left(\frac{2\pi i t}{T_m}\right)\right) \end{aligned} \quad (1)$$

As shown in Figure 1, the qubit pulse acts on the qubit and the measurement pulse, modified by the resulting qubit state, is read into the digitizer. One way to optimize the qubit pulse is to minimize the digitized measurement pulse's "loss function" against a target waveform and then use that information to determine the next iteration of the qubit pulse. For example, if we want to create an optimal π-pulse gate and we initialized the qubit in the 0 state, our target waveform would correspond to the qubit being in the 1 state. A standard QOC algorithm iterates over qubit pulses (iterates over j in Equation 1) until it finds a qubit pulse that minimizes the loss function between the measurement pulse and the target waveform.

The explicit form of the loss function depends on the measurement protocol. There are three standard measurement protocols: z-projection, randomized benchmarking (RB) and process tomography [9]. Usually at least two of the three protocols are used to provide independent verification of a qubit experiment result [10]. In the z-projection protocol, each measurement consists of applying the gate that is being tuned up and then measuring the qubit state along the z-axis. For example, tuning up an X90 gate against a reference X90 gate would yield the set of expected values $\langle Z_k\rangle = [1, 0.5, 0, 0.5, 1...]$ but the set of measured values $\langle\tilde{Z}_k\rangle = X_{\pi/2}(\tilde{X}_{\pi/2})^k$. The z-projection loss function L is defined as in Equation 2:

$$L = \sum_k |\langle Z_k\rangle - \langle\tilde{Z}_k\rangle| \quad (2)$$

In the RB protocol, each measurement consists of running the system through random circuits that would do nothing if the gates were perfect. The randomization depolarizes the quantum noise. In a typical single-qubit RB measurement, we choose s sequences of a random ordering of t single-qubit Clifford gates and repeat each sequence a large number of times. Before measuring the qubit at the end of each sequence, we apply the final "undo" Clifford gate to bring the qubit back into an eigenstate of $\hat{S}_z$ [11]. As the length of the sequence increases, the qubit is more likely to accumulate an error that results in a "wrong" final measurement. The loss function to be minimized could be either the average error per gate or the average fidelity of the gate. For multi-qubit experiments, RB could also use the addressability error rate, which measures error in the role of the qubit (target or control) or in the ability of the qubit to change its role, as a loss function [12].

In the process tomography protocol, we perform the gate (the process) on every pure state in our chosen basis set of density matrices that spans the dimension of our system (for one qubit, $d = 2$). We reconstruct an overall density matrix and measure its fidelity relative to the expected matrix [13]. The loss function to be minimized is this difference in fidelity. Machine learning techniques have become effective in assisting with tomography [14] [15] [16]. In practice, process tomography is the most difficult of the three protocols to implement. Its complexity scales exponentially with the number of qubits, the error bounds it can establish are only as good as the errors of reference "analysis gates" (Pauli matrices that combine to approximate the desired gate), and it cannot detect errors that compound over long sequences of gates. RB is more challenging to implement than z-projection, since Clifford gates can be difficult to engineer. The loss functions discussed in this paper are the most similar to z-projection, where measured values are directly compared to target values.

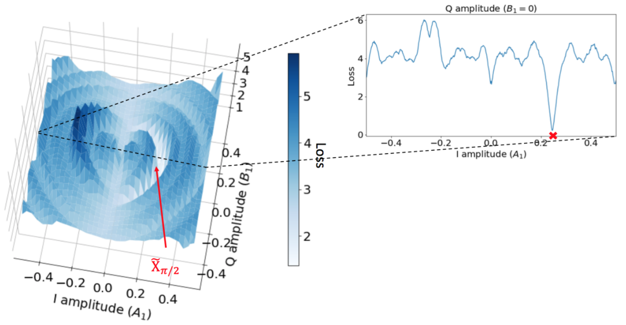

FIG. 2: The loss landscape of a two-variable $\tilde{X}_{\pi/2}$ (X90) gate (all possible initial conditions for the algorithm). The 1D projection is on the $B_1^1 = 0$ plane. The red cross represents the global minimum of the loss function. [8]

The loss landscapes discussed in this paper are n-dimensional, where I and Q each have $n/2$ Fourier components as in Equation 1. High-dimensional black-box optimization is preferred because additional Fourier components allow us to create a more precise qubit pulse.

Figure 2 shows an example of a 2D loss landscape where each signal has one Fourier component. Here we are considering all possible initial conditions ($j = 1$) for the algorithm: $I^1 = A_1^1\left(1 - \cos\left(\frac{2\pi it}{T_1}\right)\right)$ and $Q^1 = B_1^1\left(1 - \cos\left(\frac{2\pi it}{T_1}\right)\right)$. Complicated or noisy systems may have a loss landscape with many local minima. If we start from an arbitrary initial condition (an arbitrary A_1^1, B_1^1) where the loss function is not necessarily convex, it is easy to see how a QOC algorithm could get trapped in a local minimum of the loss function. Techniques have been created to remove all bad local minima from loss landscapes in fields such as machine learning [17], but until we can generalize these we have to rely on improving the robustness of QOC algorithms for this purpose (see discussion of the ADAM algorithm).

Once the problem is set up, the loss landscape is fixed. The only thing we can change is the QOC algorithm we will use to search for the global minimum of the loss function. Depending on the scenario, different algorithms will converge the fastest.

The algorithm can be open-loop, with an offline optimized control path and no intermediate feedback, or closed-loop, which utilizes intermediate feedback to iteratively optimize the control path. One example of a gradient-based open-loop QOC algorithm is Gradient Ascent Pulse Engineering (GRAPE) [18]. GRAPE is computationally expensive and so it does not scale well with system size: each iteration requires the calculation of a matrix exponential for each time step in the control pulse as well as a steepest-descent calculation. Nevertheless, various approximation methods can be employed to speed up GRAPE [19]. Gradient Optimization of Analytic Controls (GOAT), a recently proposed alternative to GRAPE, is less computationally expensive [20]. Since GRAPE is open-loop, it requires a good estimation of the system's Hamiltonian. When the system dynamics are well-understood, as was recently the case for a logical qubit encoded in an oscillator, GRAPE can be successful [10]. However, for more complicated systems, such as a Bose-Einstein condensate trapped in a magnetic micro-trap or a solid-state ensemble of coupled electron-nuclear spins, closed-loop algorithms are shown to perform better [21] [22].

Closed-loop control is ideal for black-box optimization since it does not rely on a system model and it is robust to system noise, so it scales better with system size. Gradient-based closed-loop QOC algorithms are the most common, but there are exceptions. One is Randomized Stochastic Gradient Free (RSGF), which approximates the gradient as a smooth function via a convolution [23]. Another example of a closed-loop gradient-free QOC algorithm is the Nelder Mead (NM) algorithm. In each iteration of NM, the loss function is sampled at each of the $d + 1$ vertices of a simplex. The vertex with the worst loss function value is then reflected through the centroid of the d other vertices to obtain a new sampling point. If that new sampling point has a worse loss function value than the d other vertices, the simplex is contracted and the new sampling point is chosen closer to its centroid. NM is shown to be robust compared to gradient-based closed loop QOC algorithms such as SPSA, but SPSA generally outperforms NM in higher-dimensional optimization [24]. Overall, NM is relatively easy to use and has been successfully implemented in various qubit experiments [1], notably in quantum gate tuneups [25].

For the remainder of the paper I will focus on closed-loop gradient-based QOC algorithms since they are suitable for high-dimensional black-box optimization. Gradient-based QOC algorithms are also used to implement quantum versions of machine learning tasks such as generative adversarial learning [26].

Finite Difference Stochastic Approximation (FDSA) and Simultaneous Perturbation Stochastic Approximation (SPSA) are two closed-loop gradient-descent-based algorithms for quantum control. In n-dimensional space, one iteration of FDSA involves measuring a small perturbation on each individual dimension in order to estimate the gradient at the current location. Therefore, the computation time to estimate the gradient (and therefore the algorithm convergence rate) scales linearly with system size, which becomes inefficient.

If FDSA is in n dimensions, the gradient is an n-dimensional vector and estimating the gradient takes $O(n)$ time. On the other hand, an n-dimensional iteration of SPSA involves choosing a random direction in n-dimensional space that fulfills certain conditions to establish convergence, measuring a small perturbation in that direction, and using that as the estimated gradient [27]. Therefore, SPSA takes $O(1)$ time to estimate the gradient. Interestingly, the expectation value of the bias of SPSA gradient estimation is similar to that of FDSA [28]. By using SPSA instead of FDSA, the gradient estimate quality is somewhat reduced but each iteration of the algorithm is n times faster.

SPSA is currently used in a variety of QOC applications. D-Wave Systems uses SPSA to optimize their quantum processors [29] and SPSA has been successfully implemented in a variational quantum eigensolver for molecules and quantum magnets [30]. FDSA can also be successful in cases where control bandwidth is limited so shot noise is relatively contained [22].

Here is a basic description of SPSA in n dimensions (where $Q = 0$, for simplicity). If our target waveform is the set of points $\{x_k\}$, our goal is to optimize the n-dimensional vector of I signal amplitudes $\vec{A}$ where such that the SPSA loss function L is minimized with respect to $\{x_k\}$. Similar to Equation 2, the j-th iteration of the SPSA loss function is defined as in Equation 3:

$$L(\vec{A}^j) = \sum_k |x_k - \tilde{x}_k(\vec{A}^j)| \qquad (3)$$

Next, we estimate $\vec{G}(\vec{A}^j)$, the derivative of L with respect to $\vec{A}^j$. In component form,

$$G(\vec{A}^j)_i = \frac{L(\vec{A}^j + c\hat{\Delta}^j) - L(\vec{A}^j - c\hat{\Delta}^j)}{2c\hat{\Delta}^j_i} \qquad (4)$$

where the direction $\hat{\Delta}^j$ is randomly chosen in n-dimensional space and c is a constant. Finally, we use $\vec{G}(\vec{A}^j)$ to determine the amplitude vector of the next iteration of I:

$$\vec{A}^{j+1} = \vec{A}^j - \alpha^j \vec{G}(\vec{A}^j) \qquad (5)$$

where hyperparameter $\alpha^j = a/j$ is the step size or "learning rate" for a constant a and for total number of iterations j. We tune α such that the algorithm will converge after a certain number of iterations. Intuitively, if we have chosen a $\vec{A}$ such that L is minimized, any perturbation will increase L approximately equally, so the gradient $\vec{G}$ should be small.

There are a few problems with gradient-descent-based algorithms. One problem is that highly curved loss landscapes substantially slow down gradient descent. This motivated the recent proposal of a QOC version of Newton's method, which uses local curvature information to correct the gradient step size [31]. The other problem is that gradient descent algorithms easily get trapped in local minima of the loss function. Naively increasing the learning rate α might cause the algorithm to skip over some local minima but this creates the larger problem of inefficiency due to overshooting. In order to escape the local minima, α should be small for the dimensions where the algorithm is experiencing small oscillations (while trapped in local minima) and α should be large for the dimensions where the algorithm is not trapped. The Adaptive Moment Estimation (ADAM) algorithm addresses this problem by using the first and second moments of the gradient to add a momentum term which determines the descent direction and to adapt the learning rate α for each iteration of the algorithm [32]. ADAM has become a popular deep learning algorithm and its convergence for non-convex problems is actively being studied [33]. ADAM is also applied to quantum machine learning problems [34] [35].

One idea that is currently being explored in the Houck lab is to create a QOC ADAM algorithm that uses SPSA to do the gradient descent [36]. Here is a basic description of this SPSA-ADAM algorithm in n dimensions (where $Q = 0$, for simplicity). As in Equations 3 and 4, for the j-th iteration of the algorithm we compute $\vec{G}(\vec{A}^j)$. Now we compute $\vec{M}^j$, the exponentially weighted moving average of $\vec{G}(\vec{A}^j)$ (its first moment):

$$\vec{M}^j = \beta_1 \vec{M}^{j-1} + (1-\beta_1)\vec{G}(\vec{A}^j) \tag{6}$$

where $0 < \beta_1 < 1$ and $\vec{M}^0 = \vec{0}$. To correct for the bias of the initial condition we scale the moment to be $\vec{\mathcal{M}}^j$:

$$\vec{\mathcal{M}}^j = \frac{\vec{M}^j}{1-(\beta_1)^j} \tag{7}$$

$\vec{\mathcal{M}}^j$ can be thought of as an approximate average over the past $\approx 1/(1-\beta_1)$ gradients computed. As β_1 tends to 1, $\vec{\mathcal{M}}^j$ becomes a smoother function but the tradeoff is that it adapts more slowly to changes in $\vec{G}(\vec{A}^j)$. If we were to stop here and update $\vec{A}^{j+1} = \vec{A}^j - \alpha^j \vec{\mathcal{M}}^j$, the gradient average would in general cancel out small oscillations in some dimensions while accumulating net motion in others. This is known as "adding momentum" since we can re-interpret Equation 6 by loosely thinking of $\vec{M}^j$ as a velocity, $\vec{G}(\vec{A}^j)$ as an acceleration and β_1 as a friction parameter.

The final component to ADAM is the RMSprop algorithm, which uses the gradient's first and second moments to slow down α in the dimensions where there are small oscillations but speed it up in the dimensions where there is net motion. We compute $\vec{V}^j$, the exponentially weighted moving average of the element-wise square of $\vec{G}(\vec{A}^j)$ (its second moment), and bias-correct that to $\vec{\mathcal{V}}^j$ since, as with $\vec{M}^j$, $\vec{V}^0 = \vec{0}$:

$$\begin{aligned} \vec{V}^j &= \beta_2 \vec{V}^{j-1} + (1-\beta_2)\vec{G}(\vec{A}^j)^2 \\ \vec{\mathcal{V}}^j &= \frac{\vec{V}^j}{1-(\beta_2)^j} \end{aligned} \tag{8}$$

We now update $\vec{A}^j$:

$$\vec{A}^{j+1} = \vec{A}^j - \left(\frac{\alpha^j}{\sqrt{\vec{\mathcal{V}}^j} + \epsilon}\right)\vec{\mathcal{M}}^j \tag{9}$$

where a small, constant ϵ ensures that the modified learning rate denominator is not zero, and the square root is taken element-wise. If $\vec{G}(\vec{A}^j)_i$ is large, the modified learning rate slows down since the algorithm could be experiencing small oscillations in dimension i.

ADAM's hyperparameters are α, β_1, β_2 and ϵ. One usually chooses values for β_1, β_2 and ϵ and then tunes α by sweeping over some range of values. In machine learning applications of ADAM, some standard values are $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ [32].

FPGAs and Systems on a Chip (SoCs, which combine an FPGA with a CPU) are increasingly being used in quantum experiments to enable more complex signal processing, speed up data acquisition and implement real-time feedback that informs state preparation and allows for more careful study of quantum trajectories and decoherence [37]. In 2016 Yale introduced a card capable of controlling multiple qubits that integrated digitizer, demodulator, state estimator and AWG functionality onto a single Xilinx Virtex 6 FPGA [38]. In 2017 ETH Zurich implemented FPGA-based hardware averaging and calibration such that they could discriminate between qubit ground and excited states within 50 ns with over 98% fidelity [39]. In 2018, USTC implemented a neural network that mediated active reset on a Zynq-AX7020 SoC, obtaining 99.5% readout fidelity within 171 us [34]. FPGAs are also becoming more prevalent in general quantum computing efforts beyond QOC. Delft University of Technology recently created a prototype of an FPGA-based microarchitecture for a superconducting quantum processor [40].

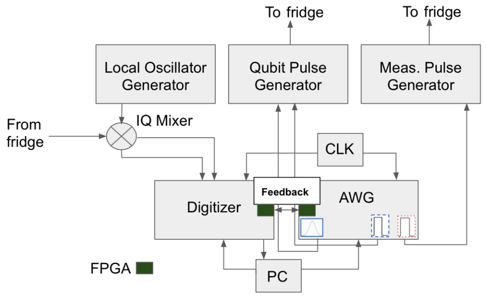

FIG. 3: Modified setup for a qubit experiment. An FPGA on the digitizer calculates the qubit state based on the digitized I and Q waveforms and transmits the state information to an FPGA on the AWG. The AWG then immediately outputs the next set of pulses to begin the next experiment. In this setup the PC's main function is to store digitizer data. In this modified setup, latency between measurements is significantly reduced.

Integrating FPGAs into a qubit data acquisition system will assist QOC and quantum experiments at large. QOC algorithms such as black-box optimization will be streamlined by on-board computation, calibration, hardware averaging, and active reset leading to fast readout.

References

[1] M. Goerz, F. Motzoi, K. B. Whaley and C. P. Koch, npj Quantum Information volume **3**, Article number: 37 (2017)

[2] F. Tacchino, C. Macchiavello, D. Gerace and D. Bajoni, npj Quantum Information volume **5**, Article number: 26 (2019)

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

[4] J. M. Chow *et. al*, Phys. Rev. A **82**, 040305(R) (2010)

[5] E. Lucero *et. al*, Phys. Rev. A **82**, 042339 (2010)

[6] J. M. Martinis and M. R. Geller, Phys. Rev. A **90**, 022307 (2014)

[7] J. Raftery, A. Vrajitoarea, G. Zhang, Z. Leng, S. J. Srinivasan and A. A. Houck, arXiv:1703.00942 [quant-ph]

[8] Figure from Zhaoqi Leng's 2019 APS March Meeting slides

[9] J. M. Chow *et. al*, Phys. Rev. Lett. **102**, 090502 (2009)

[10] R. W. Heeres *et al.*, Nature Communications volume **8**, Article number: 94 (2017)

[11] E. Knill *et al.*, Phys. Rev. A **77**, 012307 (2008)

[12] J. M. Gambetta *et. al*, Phys. Rev. Lett. **109**, 240504 (2012)

[13] T. Hazard, "Improving Quantum Hardware: Building New Superconducting Qubits and Couplers", Ph.D. thesis, Princeton University (2019)

[14] G. Carleo *et. al*, arXiv:1903.10563 [comp-ph]

[15] C. A. Ryan *et. al*, Phys. Rev. A **91**, 022118 (2015)

[16] J. Carrasquilla, G. Torlai, R. G. Melko and L. Aolita, Nature Machine Intelligence volume **1**, pages 155161 (2019)

[17] J. Sohl-Dickstein and K. Kawaguchi, arXiv:1901.03909 [stat]

[18] QuTiP documentation on the GRAPE algorithm `http://qutip.org/docs/4.0.2/guide/guide-control.html#the-grape-algorithm`

[19] G. Bhole and J.A. Jones, Front. Phys. (2018) 13: 130312.

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

[21] J.J. Sorensen, M. Aranburu, T. Heinzel, J. Sherson, arXiv:1802.07509 [quant-ph]

[22] G. Feng *et. al*, Phys. Rev. A **98**, 052341 (2018)

[23] S. Ghadimi and G. Lan, Society for Industrial and Applied Mathematics J. Optim., 23(4), pages 2341-2368 (2013)

[24] N. Dong, D. J. Eckman, M. Poloczek, X. Zhao and S. G. Henderson, arXiv:1705.078252 [math.OC] (Published in 2017 Winter Simulation Conference (WSC))

[25] M. A. Rol *et. al*, Phys. Rev. Appl. **7**, 7, 041001 (2017)

[26] L. Hu *et. al*, Science Advances Vol. **5**, no. 1, eaav2761 (2019)

[27] J.C. Spall, IEEE Transactions on Automatic Control, Volume **37**, Issue 3 (1992)

[28] J.C. Spall, Johns Hopkins APL Technical Digest, Volume **19**, Number 4 (1998)

[29] G. Quiroz, arXiv:1710.05972 [quant-ph] (Accepted to PRA on April 19, 2019)

[30] A. Kandala *et. al*, Nature **549**, pages 242-246 (2017)

[31] P. Rebentrost, M. Schuld, L. Wossnig, F. Petruccione and S. Lloyd, arXiv:1612.01789 [quant-ph]

[32] D. P. Kingma and J. Ba, arXiv:1412.6980 [cs.LG] (Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015)

[33] X. Chen, S. Liu, R. Sun and M. Hong, arXiv:1808.02941 [cs.LG] (Published as a conference paper at ICLR 2019)

[34] Z. Ding *et. al*, arXiv:1810.07997 [quant-ph] (Under submission at IEEE Transactions on Instrumentation and Measurement)

[35] M. Y. Niu, S. Boixo, V. N. Smelyanskiy H. Neven, npj Quantum Information volume **5**, Article number: 33 (2019)

[36] Z. Leng *et. al*, arXiv:1910.03591 (2019)

[37] D. Riste and L. DiCarlo, arXiv:1508.01385 [quant-ph] (chapter to appear in "Superconducting Devices in Quantum Optics)

[38] Y. Liu, "Quantum Feedback Control of Multiple Superconducting Qubits", Ph.D. thesis, Yale University (2016)

[39] T. Walter *et. al*, Phys. Rev. Appl. **7**, 054020 (2017)

[40] X. Fu *et. al*, MICRO-50 '17 Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture, Pages 813-825