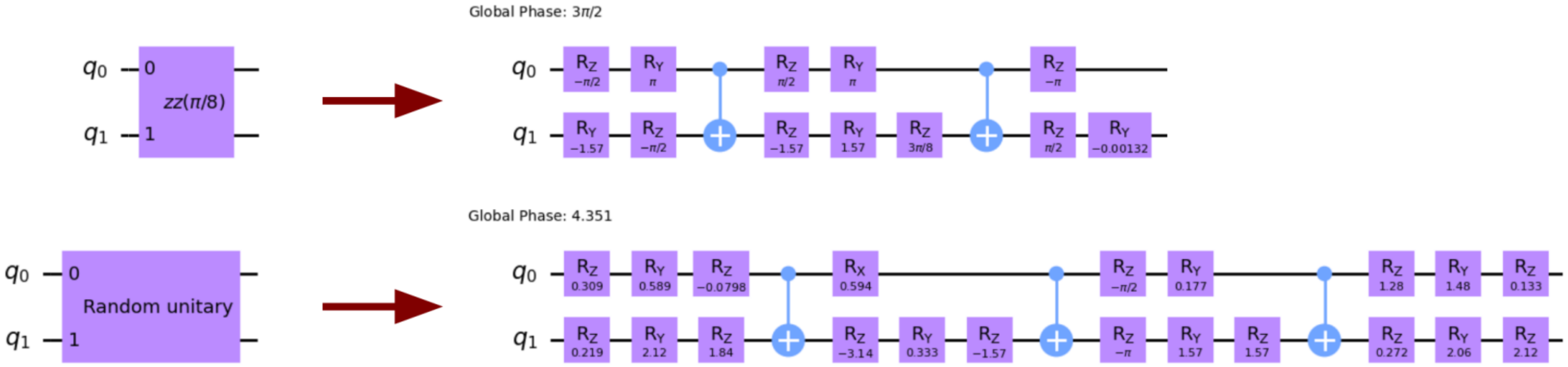
Efficient control pulses for continuous quantum gate families through coordinated re-optimization

Jason Chadwick and Fred Chong

University of Chicago

Motivation

- Hardware typically only supports one specific two-qubit operation

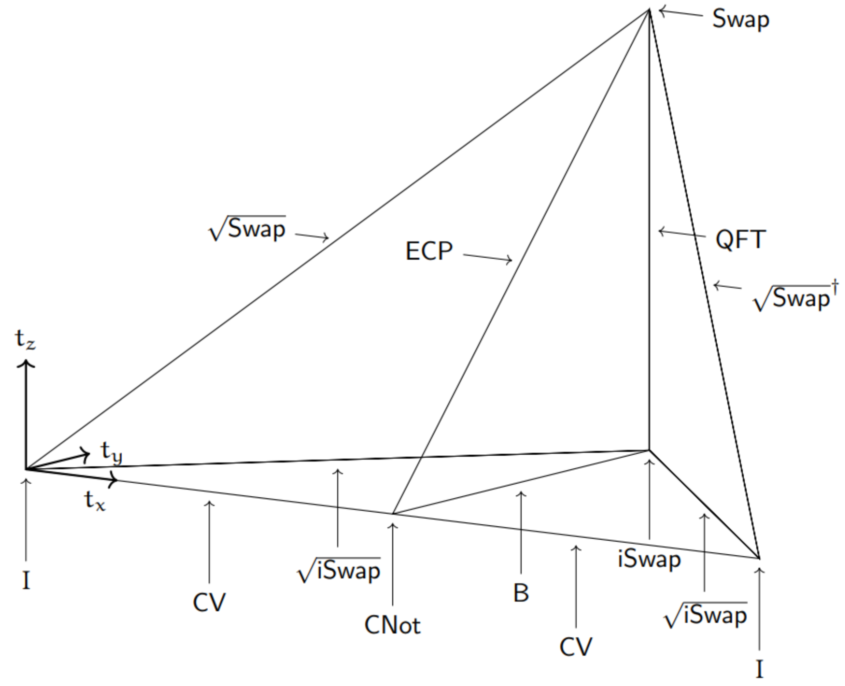


- What if every two-qubit operation was implemented natively in a single pulse sequence?
- But we don't want to optimize pulses at runtime

Background: the Weyl chamber

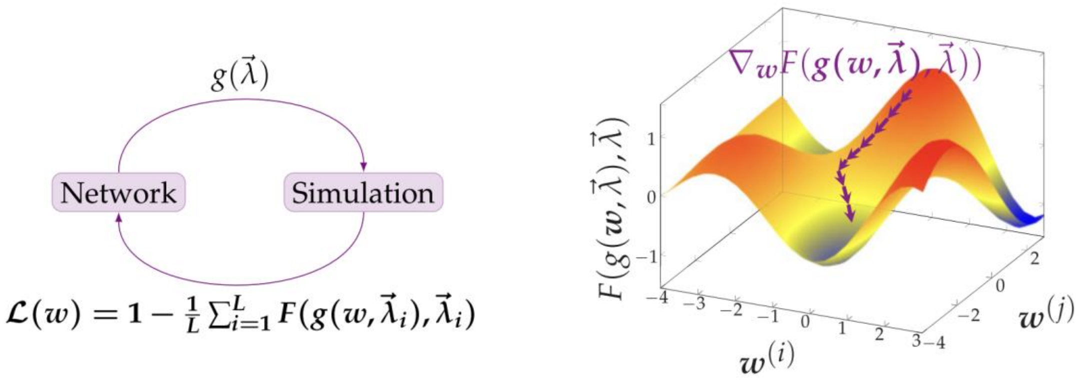
- Any two-qubit gate can be specified (up to single-qubit corrections) with three parameters
- The Weyl chamber contains the parameters of all two-qubit gates

$$U = k_1 \exp\left(-i\frac{\pi}{2} \sum_{j=x,y,z} t_j \sigma_j^{(1)} \sigma_j^{(2)} \right) k_2$$

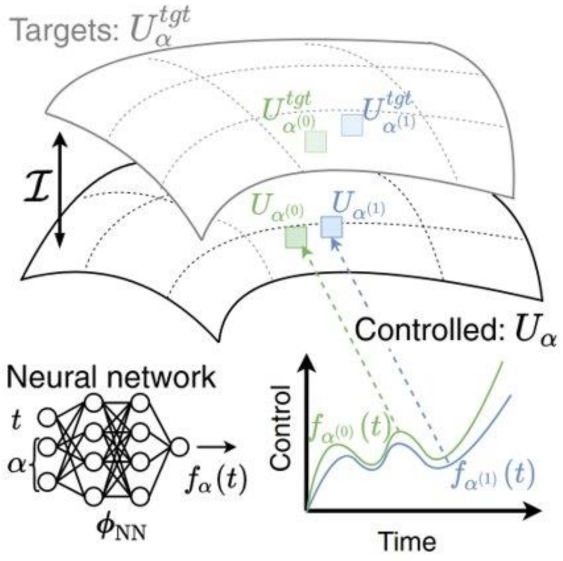


Previous work: neural networks

- Idea: train neural network to output pulse for given gate parameters
- Downside: not very flexible



Preti et al., PRX Quantum 3, 040311 (2022)

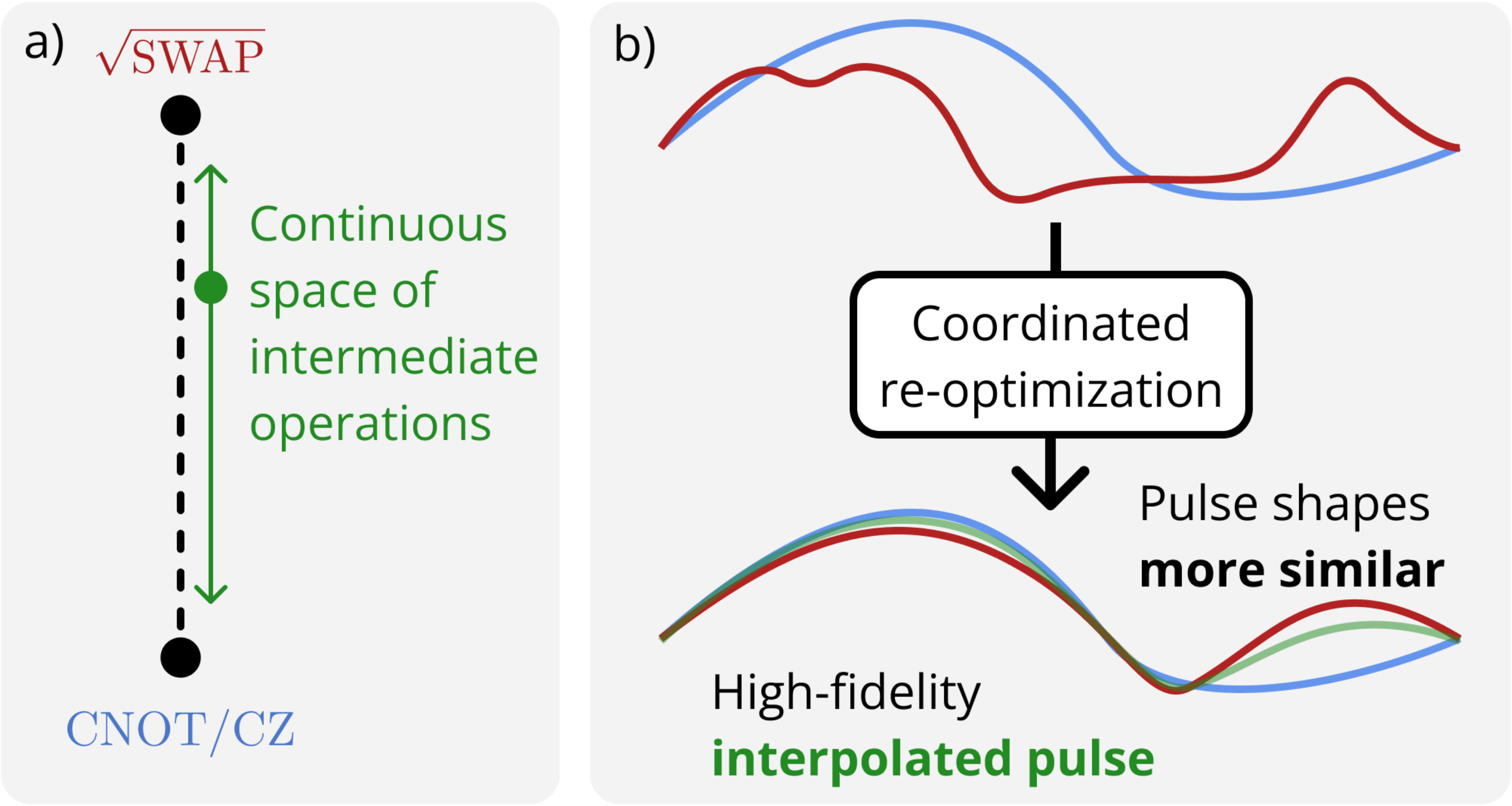


Sauvage and Mintert, Phys. Rev. Lett. 129, 050507 (2022)

Can we find a way to do this while leveraging quantum optimal control techniques?

- *Many* techniques for quantum optimal control have been developed for different scenarios and purposes (open loop, closed loop, RL, trajectory optimization, …)
- Neural network-based methods miss out on all these optimizations
- Can we obtain similar results while retaining the benefits of advanced pulse optimization techniques?

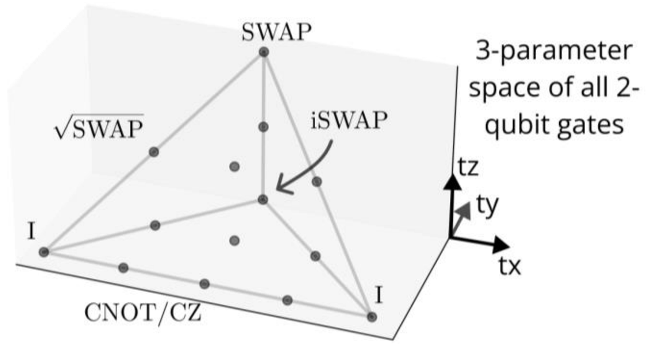
Idea: specifically design pulses for interpolation



arxiv:2302.01553

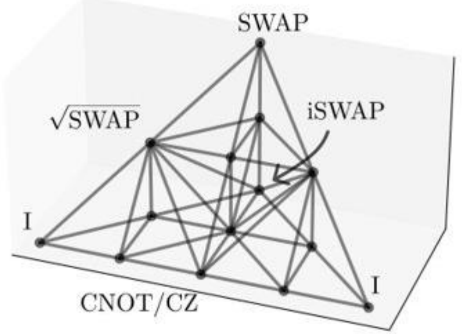
Example: all two-qubit gates

Initial pulse seeding



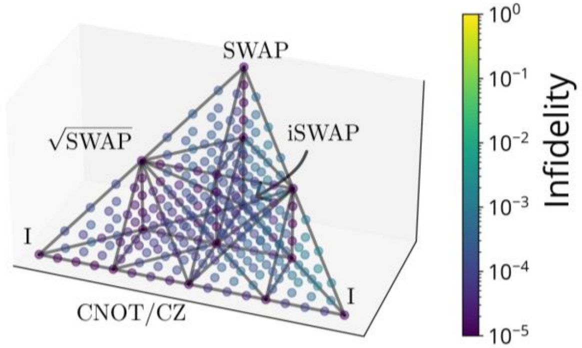
- Identify **reference points** in parameter space
- Optimize control pulses for each reference point

Reference pulse re-optimization



- Create **simplicial mesh**
- **Re-optimize** reference pulses based on neighbors

Interpolation on calibrated landscape



- Use **linear interpolation** to obtain pulses for any point in the continuous gate set

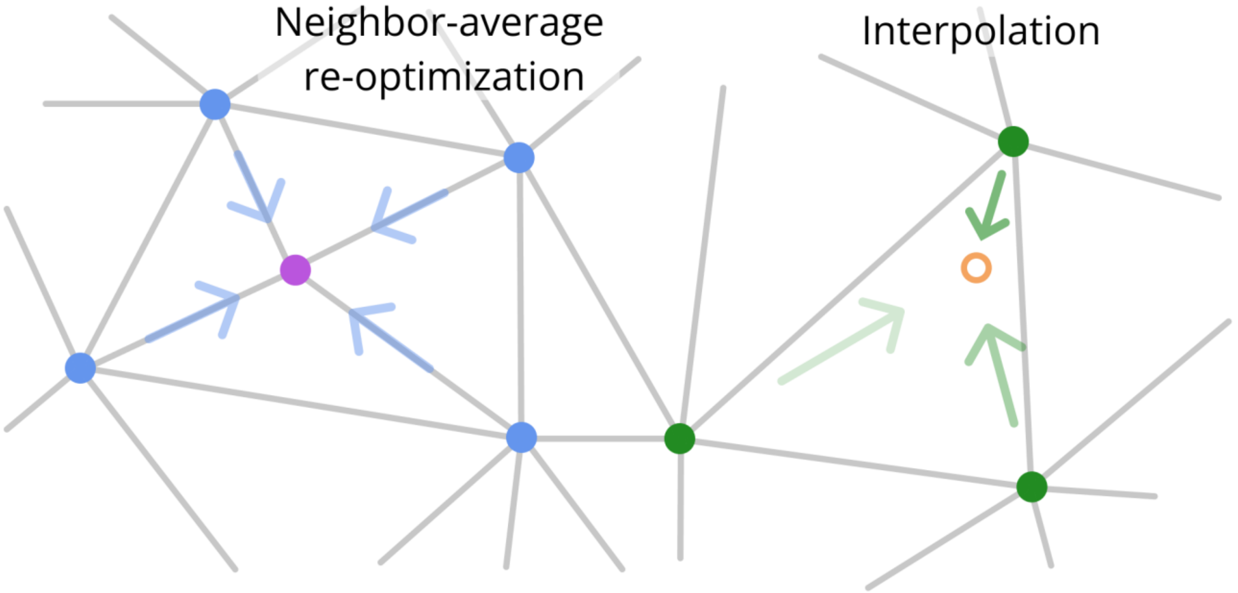
arxiv:2302.01553

Details: re-optimization and interpolation

$$\hat{\vec{\alpha}}_i = \frac{1}{|\eta(p_i)|} \sum_{p_j \in \eta(p_i)} \vec{\alpha}_j$$

$$J = \underbrace{1 - \frac{1}{h^2}\left|\mathrm{Tr}\left(U^\dagger_{\mathrm{target}} U_T\right)\right|^2}_{\text{gate infidelity}} + \underbrace{\tilde{\lambda} \sum_k^{n_f} \|\vec{\alpha}_k - \vec{\alpha}_{0,k}\|_2^2}_{\text{Tikhonov regularization}}$$

1 *round* of re-optimization: do this sequentially for every reference point



$$\hat{\vec{\alpha}}_p = \frac{1}{4}\left(\vec{\alpha}_0 + \vec{\alpha}_1 + \vec{\alpha}_2 + \vec{\alpha}_3\right)$$

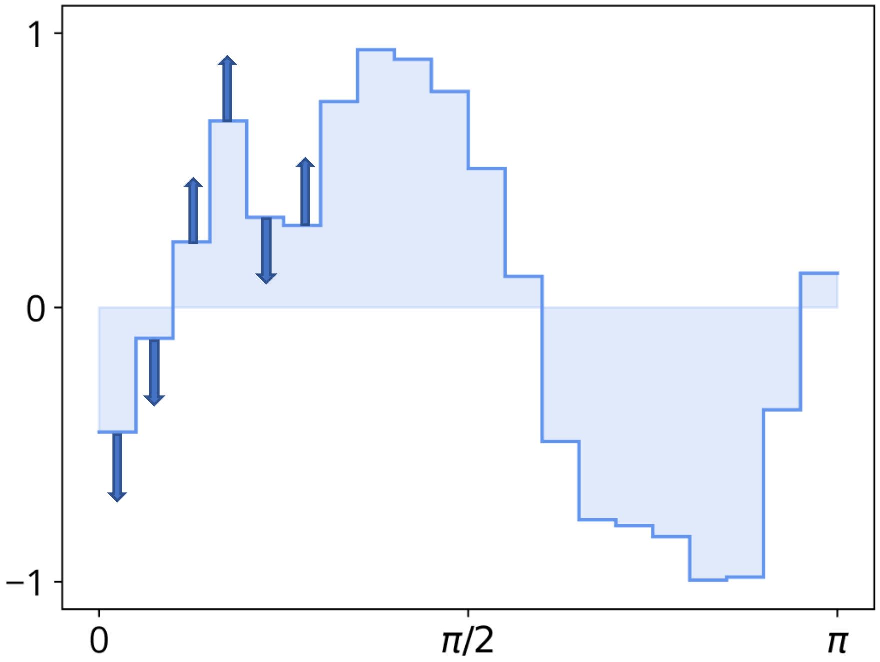
$$\vec{\alpha}_{\tilde{p}} = 0.73\vec{\alpha}_0 + 0.11\vec{\alpha}_1 + 0.16\vec{\alpha}_2$$

$$\vec{\alpha}_{\tilde{p}} = \sum_{p_i \in S_{\tilde{p}}} b_i \vec{\alpha}_i$$

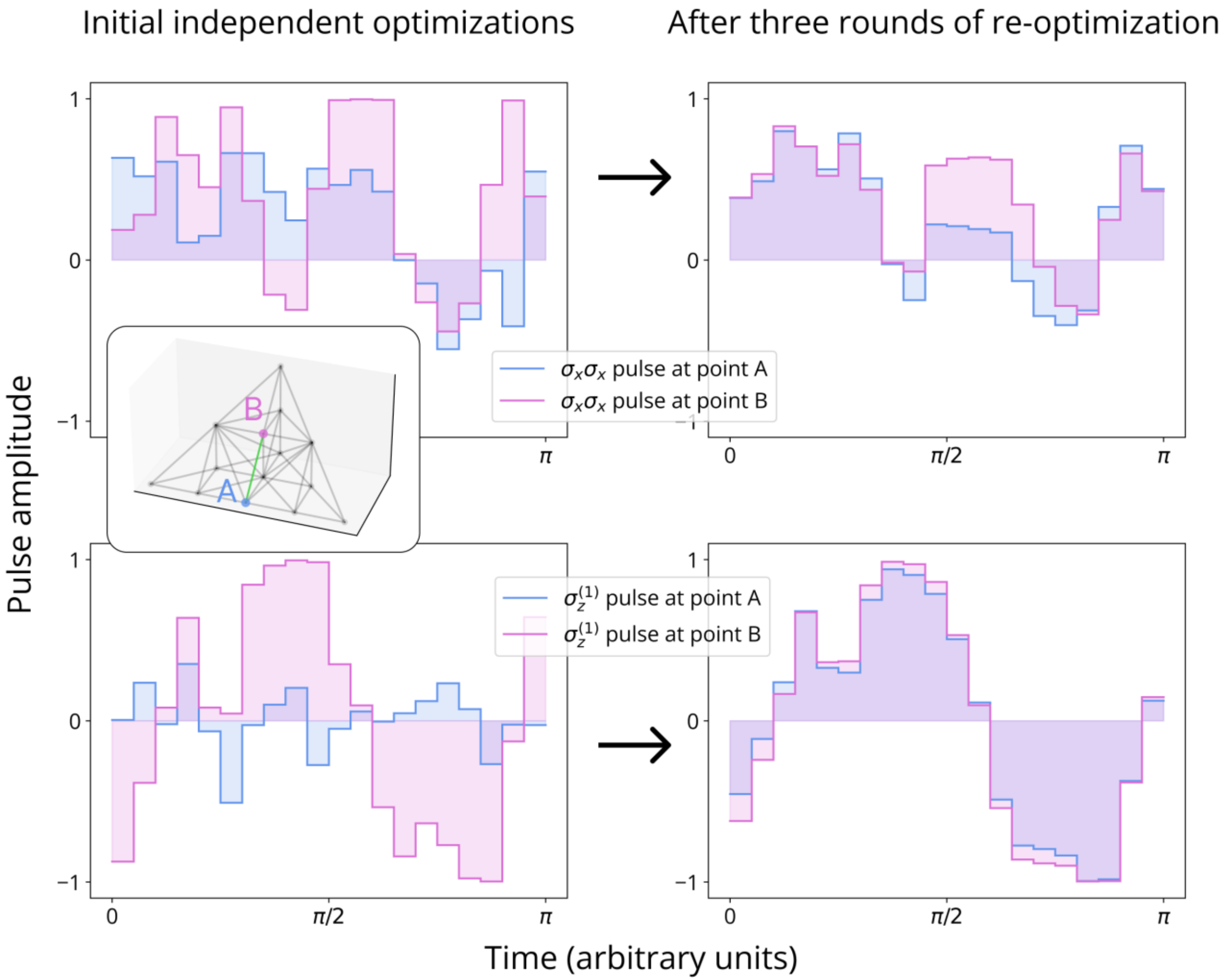
arxiv:2302.01553

Example: simple two-qubit Hamiltonian

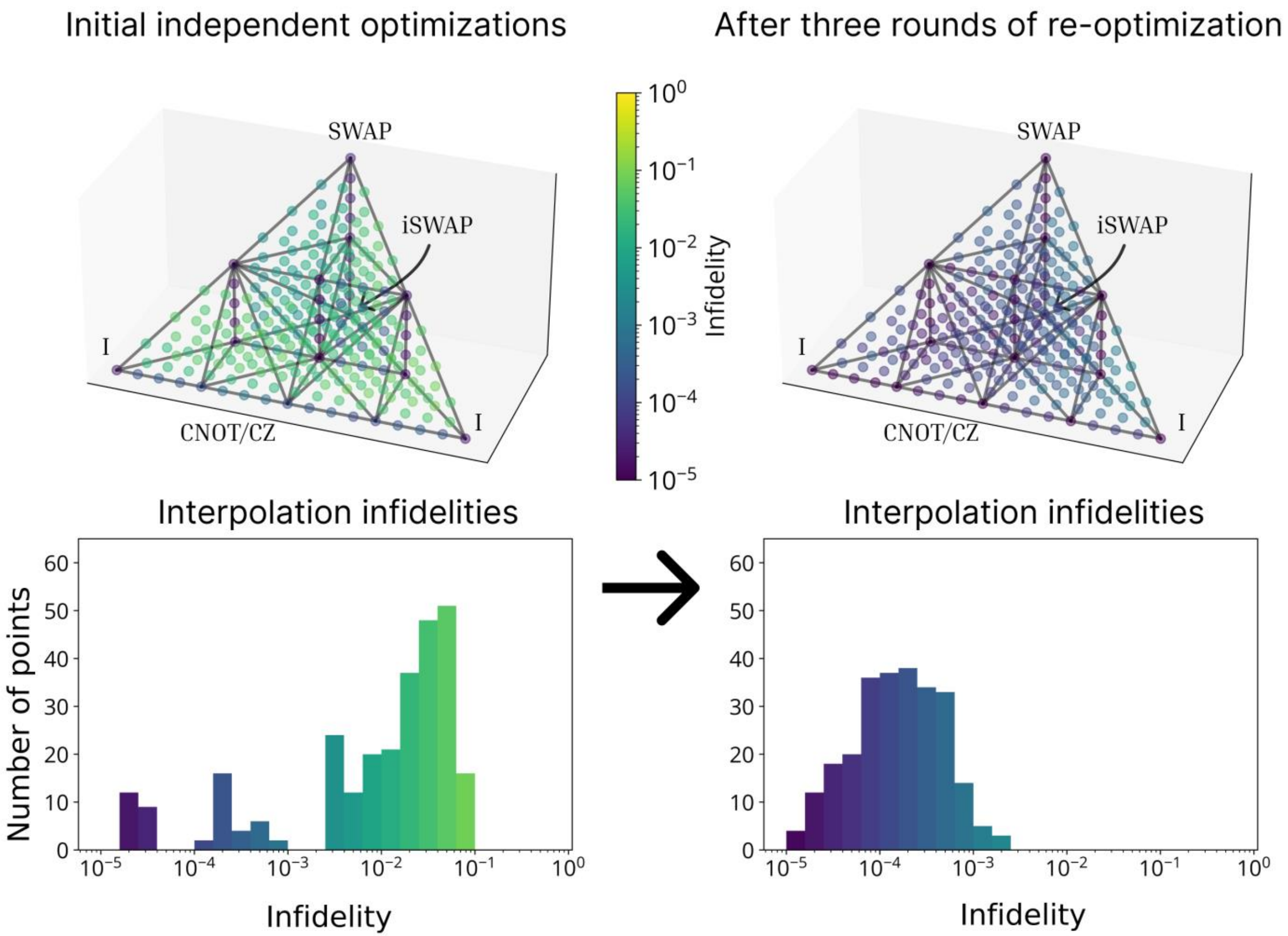
$$H(t) = f^{\vec{\alpha}}_{xx}(t)\sigma^{(1)}_x\sigma^{(2)}_x + \sum_{j=1}^{2} f^{\vec{\alpha}}_{jy}(t)\sigma^{(j)}_y + f^{\vec{\alpha}}_{jz}(t)\sigma^{(j)}_z$$



Re-optimization makes pulses for nearby reference points look similar

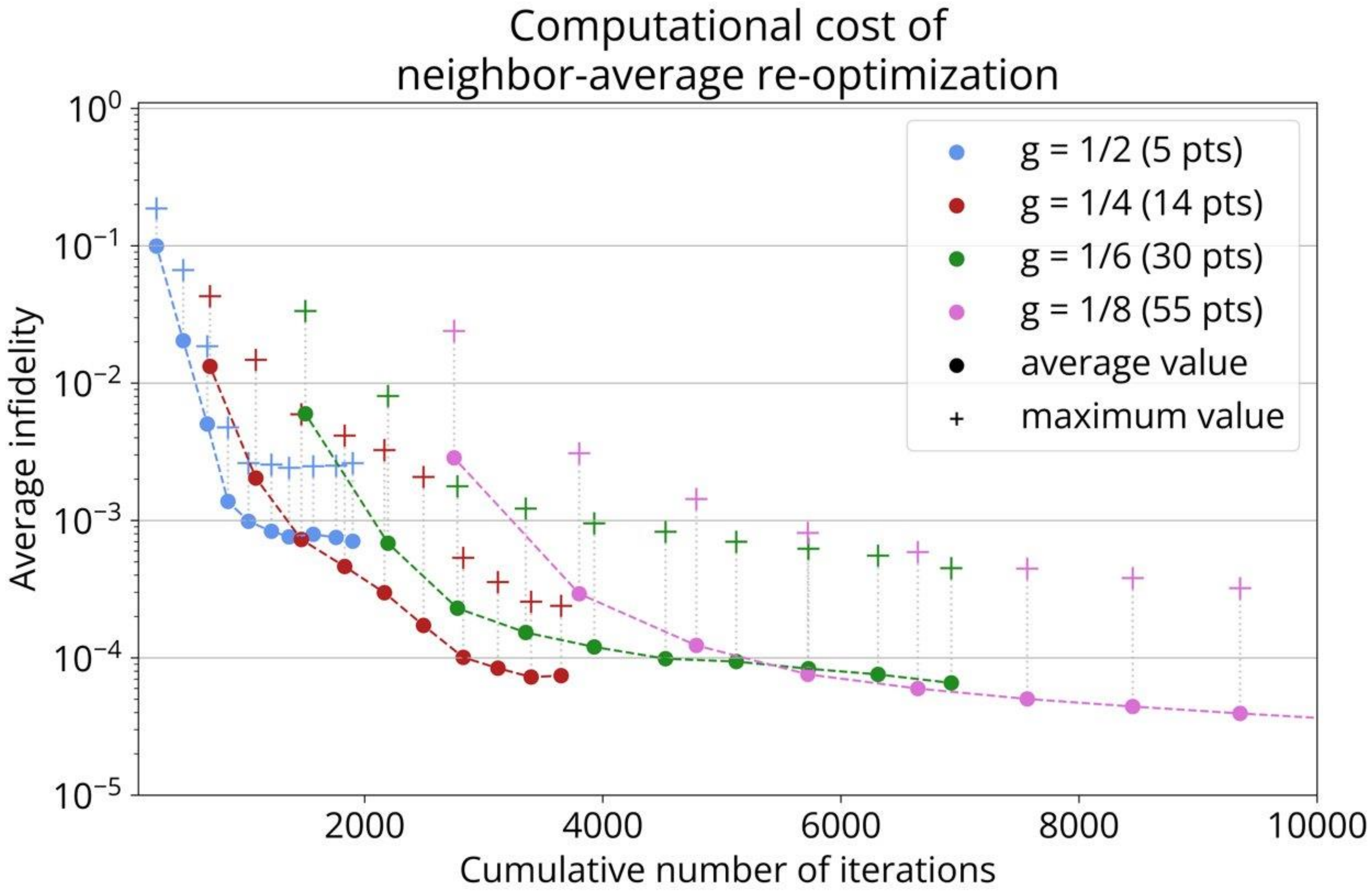


Re-optimization improves interpolations



Calibration time vs. performance

- Tradeoff between interpolation quality and classical computation time
- Each round of re-optimization adds classical computation cost
- Number of reference points adds computation cost


Computational cost of
neighbor-average re-optimization
Average infidelity
Cumulative number of iterations
g = 1/2 (5 pts)
g = 1/4 (14 pts)
g = 1/6 (30 pts)
g = 1/8 (55 pts)
average value
maximum value
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
2000
4000
6000
8000
10000

Comparison to neural network approaches

- Previous work[1,2]: neural network for gate-family pulse generation
  - Input: parameters of gate; time t
  - Output: Control pulse values at time t
- For the same gate family, we use **2x less computation** (or better) for a lower average pulse infidelity
- We have additional benefits of explainability and modularity

[1] Sauvage and Mintert, "Optimal Control of Families of Quantum Gates," PRL 129, 050507 (2022)

[2] Preti, Calarco, and Motzoi, "Continuous Quantum Gate Sets and Pulse-Class Meta-Optimization," PRX Quantum 3, 040311 (2022)

Modularity: pulse optimizer

- Any pulse optimization method can be used in this framework; just need Tikhonov regularization in cost function
- Offline, model-based optimization may not be enough on noisy devices
- **Data-driven optimization** can solve device-model mismatch

Modularity & extensions

- Linear **interpolation method** is likely not the best choice
- **Reference point distribution** can be changed
- Neighbor-average **re-optimization method** can be changed
  - Selective re-optimization
- **Pulse parameterization** can be changed to add robustness or account for device constraints
- Extension: is **recalibration** under device drift more efficient?
- Maybe a smaller subset of Weyl chamber is enough for significant performance improvements on most circuits

Summary

- We provide a method to calibrate a small number of control pulses for high-quality **interpolation**
- After an initial calibration, our method instantly generates high-fidelity control pulses for arbitrary gates in the chosen continuous set
- We improve on previous neural network methods by **reducing computation time** and improving **explainability**
- The method is **modular** - can use advanced optimizers or make other tweaks to method

Extension idea: parameterized Hamiltonian

$$H(t) = \sum_i \omega_i a_i^\dagger a_i + \frac{\Delta_i}{2} a_i^\dagger a_i^\dagger a_i a_i + f_x^i(t)(a_i^\dagger + a_i)$$
$$+ \sum_j \sum_{i<j} J_{ij}(a_i^\dagger a_j + a_i a_j^\dagger)$$
$$+ \ldots$$

arxiv:2302.01553

Extension idea: parameterized Hamiltonian

- Goal: perform a good CNOT operation for **any** specific instance of the more general Hamiltonian
- Proposed method:
  - **Generate pulse interpolation landscape** for parameterized Hamiltonian (using similar methods to those described here)
  - **Characterize** qubit(s) of interest
  - Instantly obtain a good pulse for that specific Hamiltonian (if device-model agreement is good)
- Changes the focus from **optimization** to **characterization**
  - Allows for **more complex/expensive** pulse optimizations
- Same interpolation can work on any qubit in the device