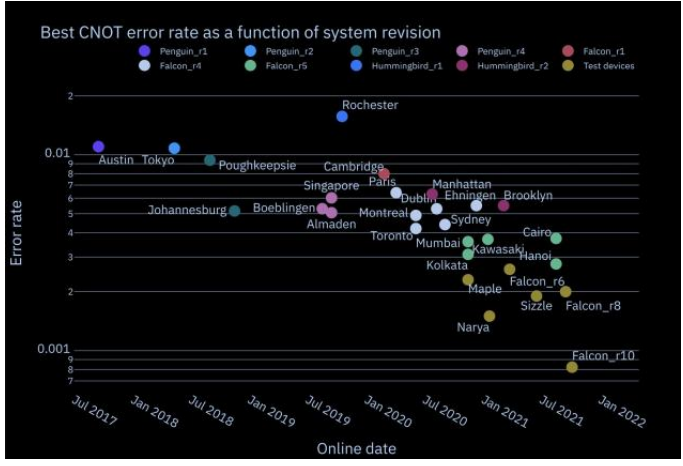
Improving Quantum Gates with Optimal Quantum Control

Luciano Pereira

It is necessary to improve the fidelity of quantum gates to achieve computational advantage with quantum computers.



https://twitter.com/jaygambetta/status/1445115380616335373

An alternative is use optimal quantum control.

$\rightarrow$ Gradient Ascent Pulse Engineering (GRAPE) algorithm.

Let us consider a system described by the Hamiltonian $H(t) = H(w(t))$, where $w(t)$ are continuous control parameters. GRAPE divides the temporal evolution into small temporal pieces with constant control parameters,

$$H(t_k) = H_k(w_k).$$

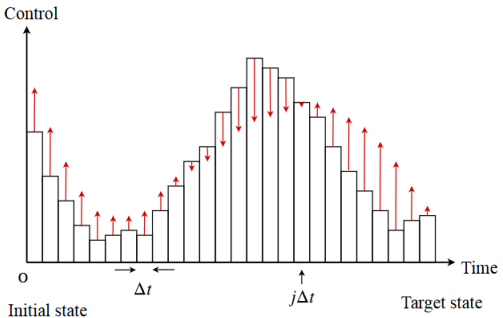
Then, considering a time step Δt_k, we have that the evolution piece is

$$U_k(w_k) = e^{-iH_k(w_k)\Delta t_k}.$$

After N time steps, the total evolution is

$$U(t_k, \vec{w}) = U_k(w_k)U_{k-1}(w_{k-1}) \cdots U_1(w_1)U_0(w_0),$$

where $\vec{w}$ is a vector with the control parameters.



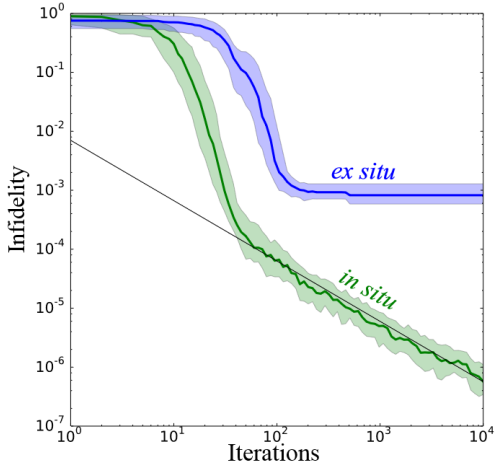
Y. Shi *et al.*, "Optimized Compilation of Aggregated Instructions for Realistic Quantum Computers".

In order to find the optimal control parameters we minimize numerically the fidelity with respect to the target gate V,

$$f(\vec{w}) = \frac{1}{d^2} |Tr(V^\dagger U(t_k, \vec{w}))|^2.$$

The performance of this quantum control depends on the precision of the model. We call it *ex-situ* quantum control.

We can consider an alternative quantum control where the fidelity is evaluated on the quantum device, so that there are no limitations due to errors in the modeling. We call it *in-situ* quantum control or *quantum feedback*.



C. Ferrie and O. Moussa, "Robust and efficient in situ quantum control".

There exist experimental techniques to evaluate the fidelity between an experimental gate and a theoretical gate.

- Randomize benchmaking.
- Direct fidelity estimation

  S. T. Flammia and Y.-K. Liu, "Direct Fidelity Estimation from Few Pauli Measurements".

- Build a Qiskit library to perform the ex-situ and in-situ quantum control with GRAPE.
- Propose a mixed protocol, where first the ex-situ quantum control is carried out, to then refine the result with in-situ quantum control.
- Implement some relevant gate, such as c-not or toffoli.

Qiskit pulse has all the needed to carry out this project.

hamiltonian

$$
\begin{aligned}
\mathcal{H}/\hbar = & \sum_{i=0}^{6}\left(\frac{\omega_{q,i}}{2}(\mathbb{I}-\sigma_i^z)+\frac{\Delta_i}{2}(O_i^2-O_i)+\Omega_{d,i}D_i(t)\sigma_i^X\right) \\
& + J_{0,1}(\sigma_0^+\sigma_1^- + \sigma_0^-\sigma_1^+) + J_{1,2}(\sigma_1^+\sigma_2^- + \sigma_1^-\sigma_2^+) + J_{4,5}(\sigma_4^+\sigma_5^- + \sigma_4^-\sigma_5^+) + J_{5,6}(\sigma_5^+\sigma_6^- + \sigma_5^-\sigma_6^+) \\
& + J_{1,3}(\sigma_1^+\sigma_3^- + \sigma_1^-\sigma_3^+) + J_{3,5}(\sigma_3^+\sigma_5^- + \sigma_3^-\sigma_5^+) \\
& + \Omega_{d,0}(U_0^{(0,1)}(t))\sigma_0^X + \Omega_{d,1}(U_1^{(1,0)}(t) + U_3^{(1,3)}(t) + U_2^{(1,2)}(t))\sigma_1^X \\
& + \Omega_{d,2}(U_4^{(2,1)}(t))\sigma_2^X + \Omega_{d,3}(U_5^{(3,1)}(t) + U_6^{(3,5)}(t))\sigma_3^X \\
& + \Omega_{d,4}(U_7^{(4,5)}(t))\sigma_4^X + \Omega_{d,5}(U_8^{(5,3)}(t) + U_{10}^{(5,6)}(t) + U_9^{(5,4)}(t))\sigma_5^X \\
& + \Omega_{d,6}(U_{11}^{(6,5)}(t))\sigma_6^X
\end{aligned}
$$

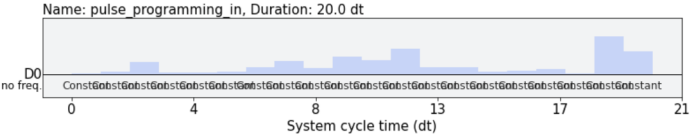
[54]:
```
from qiskit import pulse
import numpy as np

dc = pulse.DriveChannel
```

[55]:
```
Amp = abs(np.random.randn(20))
Amp = Amp/max(Amp)
dt = 1
```

[58]:
```
with pulse.build(name='pulse_programming_in') as pulse_prog:
    for amp in Amp:
        pulse.play( pulse.library.Constant(dt, amp, name=None), dc(0) )
```

[59]:
```
pulse_prog.draw()
```

[59]:



Biggest challenge: evaluate the fidelity experimentally.

I already have a module to measure the fidelity, but only between states.

We have to extend it to gates.

```
def Direct_Fidelity_Estimation( ρ, ψ, M, *args  ):
    """
    Direct Fidelity Estimation (DFE).
    Estimate the fidelity <ψ|ρ|ψ> measuring M Pauli Operators.

    Inputs:
        ρ     : 2^n qubits Density Matrix. Can be a qiskit circuit.
        ψ     : 2^n qubits Pure state.
        M     : Number of observables for the DFE.
        *args : Extra inputs of the fun function.
    Output:
        Fid : Estimated Fidelity.
    """

    if isinstance(ρ, QuantumCircuit):
        fun = Expected_Value_Qiskit
    else:
        fun = None

    n = int( np.log2(ψ.size) )
    σ = np.outer( ψ, ψ.conj() )
    paulis = np.array( [ [1,0,0,1], [0,1,1,0],[0,-1j,1j,0],[1,0,0,-1] ] ).reshape(4,2,2) / np.sqrt(2)
    Pk = n*[ paulis ]
    σk = np.real( InnerProductMatrices( σ, Pk ) ).flatten()
    qk = np.real(σk**2)
    Index = rm.choices( range(4**n), qk, k = M )
```