QOC with A Single NV Center

Hendry

February 23, 2025

Qubit's Hamiltonian

In the qubit's rotating frame,

$$\hat{H} = \frac{\delta(t)}{2}\hat{\sigma}_z + \frac{\Omega_1}{2} f(t)\,[1+\epsilon(t)]\,[\hat{\sigma}_x \cos(\phi(t)) + \hat{\sigma}_y \sin(\phi(t))]$$

where

$$\begin{aligned} \Omega_1 f(t) &: \text{modulated Rabi frequency} \\ \phi(t) &: \text{pulse's initial phase} \\ \delta(t), \epsilon(t) &: \text{noises} \end{aligned}$$

Physical observables are represented by

$$\overline{\rho}(t) = \frac{1}{N}\sum_n e^{-i\hat{H}_n t}\rho(0)e^{i\hat{H}_n t}$$

$\delta(t)$ as an Ornstein-Uhlenbeck Process

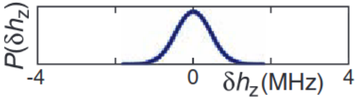


Figure: (Hanson et al., 2008)

$$\hat{H} = \frac{\delta(t)}{2}\hat{\sigma}_z + \dots$$

Fully relaxed OU process. Starting as a Gaussian distribution with mean 0 and variance σ^2, the updating formula is

$$X(t + \Delta t) = X(t)e^{-(\Delta t)/\tau} + \tilde{n}\sigma\sqrt{1 - e^{-2(\Delta t)/\tau}}$$

where

$\tilde{n}$: unit Gaussian sample

τ : relaxation time

$\epsilon(t)$ as fractional amplitude uncertainty

$$\hat{H} = \cdots + \Omega_1 f(t) \left[1 + \epsilon(t)\right] \ldots$$

Can be modeled by a fully-relaxed OU process with large τ.

Expressing σ and τ in terms of T_2^* and T_2^{HE}

$$0 = 4\tau e^{-T_2^{\mathrm{HE}}/2\tau} - \tau\left(e^{-T_2^{\mathrm{HE}}/\tau} + e^{-T_2^*/\tau} + 2\right) + T_2^{\mathrm{HE}} - T_2^*$$

$$\frac{1}{\sigma^2} = \tau^2 e^{-T_2^*/\tau} - \tau^2 + \tau T_2^*$$

credit: Pascual-Winter et al., 2012.

Coding: Generating OU processes

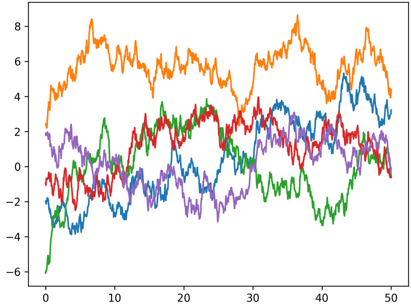
```
T2_star = 0.5
T2_Hahn = 3

delta = ou.ou_noise(T2_star=T2_star, T2_Hahn=T2_Hahn)
delta.sigma, delta.tau
```

[145] ✓ 0.0s

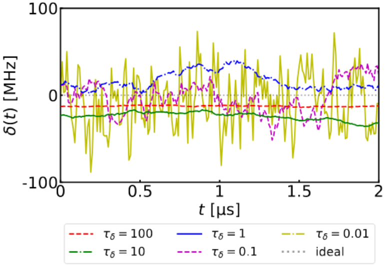
```
(np.float64(2.842272904735199), np.float64(17.0233192862102))
```

```
tlist = np.linspace(0, 50, 1001)
dt = np.diff(tlist)[0]
sample = 5

out_arr = delta.generate(tlist, sample)

plt.plot(tlist, out_arr);
```

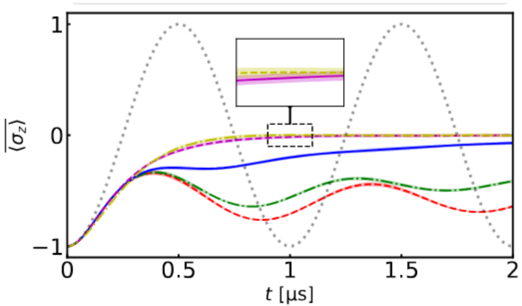
[147] ✓ 0.3s



Tuning τ for $\delta(t)$

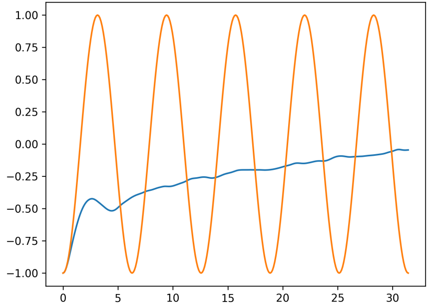


credit: Lim et al., 2024



Coding: Simulating the Rabi oscillation

```python
qb = ou.qbevo(sample = 1000,
              rho_0 = ou.bloch_state(theta = np.pi,
                                     phi = 0),
              time = ou.timeline([10*np.pi, 10*np.pi/300]),
              Omega_1 = 1,
              f = 1,
              phi = 0,
              delta = delta)

t, sz_t = qb.run(expect = [ou.pauli()[2]],
                 sample_mean = True,
                 flatten = True)

qb_ideal = qb.copy(ideal=True)

t_ideal, sz_t_ideal = qb_ideal.run(expect = [ou.pauli()[2]],
                                   sample_mean = True,
                                   flatten = True)
```



Quantum Optimal Control

Loose definition: Tuning variables in the system dynamics to achieve a certain goal.

Controllability: The pulsed NV center is **completely controllable.**

Complete controllability: A system is completely controllable if the Hamiltonian generates all possible gate operations (D'Alessandro, 2022).

$$\hat{H} = \frac{\delta(t)}{2}\hat{\sigma}_z + \frac{\Omega_1}{2} f(t)\left[1+\epsilon(t)\right]\left[\hat{\sigma}_x \cos(\phi(t)) + \hat{\sigma}_y \sin(\phi(t))\right]$$

Ingredients for QOC: the system dynamics

$$\hat{H} = \frac{\delta(t)}{2}\hat{\sigma}_z + \frac{\Omega_1}{2}{\color{red}f(t)}\left[1+\epsilon(t)\right]\left[\hat{\sigma}_x\cos(\phi(t)) + \hat{\sigma}_y\sin(\phi(t))\right]$$

f(t) is the control.

Ingredients for QOC: the control objective J

Either a **cost function** to minimize, or a **figure of merit** to maximize.

$$J_1 = 1 - \overline{F} = 1 - \frac{1}{N}\sum_n \mathrm{tr}\left(\sqrt{\sqrt{\rho_n(T)}\rho_{target}\sqrt{\rho_n(T)}}\right)$$

or

$$J_2 = 1 - \overline{F}_{\mathrm{gate}} = 1 - \frac{1}{4N}\sum_n \left|\mathrm{tr}\left(\hat{U}^{\dagger}_{\mathrm{target}}\hat{U}_n(T)\right)\right|^2$$

where

T : control duration, i.e. pulse length

credit: Rembold et al., 2020; Nielsen & Chuang, 2010

Ingredients for QOC: the control space restriction

Useful for excluding unphysical/impractical values.

Limit $\Omega_1 f(t)$ to not exceed some limit, e.g. $2\pi \times 5\ \mathrm{MHz}$, i.e. limit $-5 \leq f(t) \leq 5$.

The control landscape

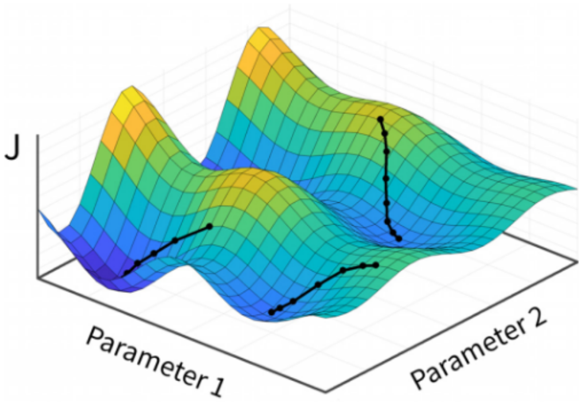


Figure: Rembold et al., 2020.

Traversing the control landscape: QOC algorithms

Type 1: **Gradient-based**

- Gradient Ascent Pulse Engineering (GRAPE).
- Krotov's method.
- Gradient Optimization Using Parameterization (GROUP).
- Gradient Optimization of Analytic Controls (GOAT).

Type 2: **Gradient-free**

- Chopped Random Basis (CRAB).
- Dressed Chopped Random Basis (dCRAB).

credit: Rembold et al., 2020.

dCRAB

- Set **superiteration** $s = 1$.
- Expand control using a **chopped basis** $\{f_k^s\}$ with **randomized basis parameters** β_k^s:

$$f^s(t) = \sum_{k=1}^{N_c} a_k^s f_k^s(\beta_k^s; t)$$

- Numerically optimize a_k^s using algorithms such as the **Nelder-Mead (NM) simplex** or **Covariance Matrix Adaptation Evolution Strategy (CMA-ES)**.
- Carry the optimized pulse in superiteration $s = 1$ over to superiteration $s = 2$. Repeat optimization.

$$f^s(t) = a_0^s f^{s-1}(t) + \sum_{k=1}^{N_c} a_k^s f_k^s(\beta_k^s; t)$$

- Rinse and repeat.

Scaling function

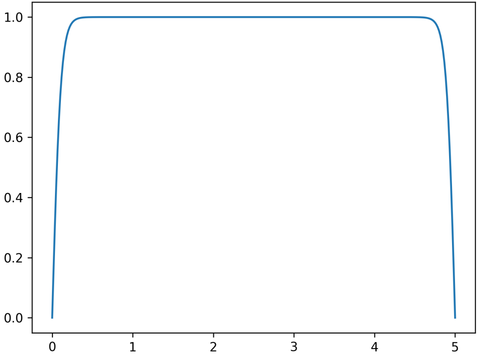
Multiplies the pulse; useful to hold the end-values at zero.

$$\Lambda(t) = \tanh\left[\sigma \sin\left(\frac{\pi t}{2T}\right)\right] \tanh\left[-\sigma \sin\left(\frac{\pi(t-T)}{2T}\right)\right]$$

where

σ : ramp steepness.

T : pulse duration.



Optimization hyperparameters

- Number N_c of basis per superiteration.
- Sampling interval $[\beta_{\min}, \beta_{\max}]$ of the randomized basis parameters β_k^s.
- Amplitude variation of a_k^s between iterations.

Coding: Specifying the control landscape

```
def infidelity(rho_T, rho_target):
    return qt.fidelity(rho_T, rho_target)

qb = ou.qbevo(sample = 50,
              rho_0 = ou.bloch_state(theta = np.pi,
                                     phi = 0),
              time = ou.timeline([10*np.pi, 10*np.pi/300]),
              Omega_1 = 1,
              f = 1,
              phi = 0,
              delta = delta)

cf = ou.cost_function(qbevo = qb,
                      state_measure = infidelity)

csr = ou.control_space_restriction(lim_f = (-5, 5))

cf.pulse_index = 0
cf.calculate_original()
```

[8] ✓ 3.0s

```
... np.float64(0.6524141490725134)
```

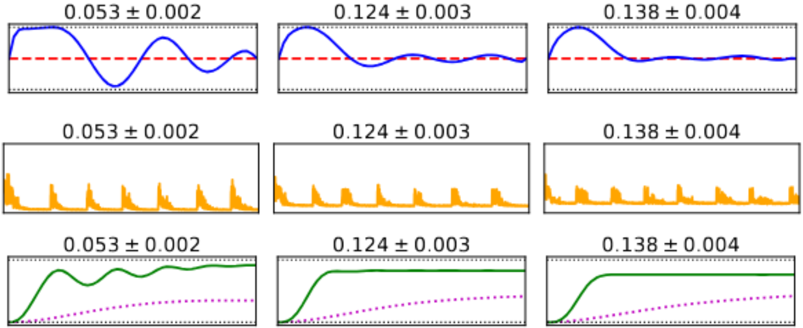
Coding: Running the optimization

```
res = ou.dCRAB(cost_function=cf,
               super_iteration=2,
               costfunc_goal=1e-3,
               p1_basis = "Fourier",
               p1_num_basis_vector = 5,
               p1_basis_parameter_lim = (1, 10))
```

✓ 1m 17.1s

```
INFO     oc_logger    The optimization direction is minimization
INFO     oc_logger    QuOCS version number: 0.0.60
INFO     oc_logger    Direct search start time has been reset.
INFO     oc_logger    New record achieved. Previous FoM: 10000000000, new best FoM : 0.5489460390149822
INFO     oc_logger    Function evaluation number: 1, SI: 1, Sub-iteration number: 0, FoM: 0.5489460390149822
INFO     oc_logger    Function evaluation number: 2, SI: 1, Sub-iteration number: 0, FoM: 0.9601123969727986
INFO     oc_logger    Function evaluation number: 3, SI: 1, Sub-iteration number: 0, FoM: 0.8710885613090092
```

Example of results



credit: Lim et al., 2024

Possible tweaks

- Consider $\phi(t)$ as well.
- Consider different bases.
- Hyperparameter tuning.