Trial and Error with CPMG and DDrf

Donghun Jung

24 Oct 2025

Department of Physics, Sungkyunkwan University

Center for Quantum Technology QuiME Lab, Korea

Institute of Science Technology

Outline

- [Quick Recap] NV system

- Machine Learning Approach: Why it fails?

- Strategy

- Implementation of Runge-Kutta 7
    - Why Euler Method fails
    - Key features of RK7 implementation

- Dynamic Decoupling & DDrf explanation

- Examples & Simulations

- TODO

NV Center System

Spatially separated magnetic dipoles interact with each other via magnetic dipolar interaction (i.e., magnetic dipole-dipole interaction), which dominates the hyperfine coupling between the NV$^-$ electron spin and surrounding ^{13}C nuclear spins.

Note: The nuclear spin precession axes depend on the NV electron spin state.

Figure. Diagram for NV center system. The orange-colored arrow denotes the applied external magnetic field. Drawn by Jiwon.

NV Center System

rightside

Hamiltonian for NV coupled with N ¹³C nuclear spins

(Electron spin in rotating frame, others are in lab frame)

Note: We use two level space in NV electrons, $m_s = 0$ state as $|0\rangle$ and $m_s = -1$ state as $|1\rangle$ state. $m_s = 1$ state is not used.

$$\mathcal{H} = \sum_{i=1}^{N} \gamma_c B_z I_z^i + \underbrace{A_{\parallel}^i S_z I_z^i + A_{\perp}^i S_z I_x}_{\text{Hyperfine Interaction Terms}}$$

Figure. Diagram for NV center system. The orange-colored arrow denotes the applied external magnetic field. Drawn by Jiwon.

NV Center System

Available Control Sources

Microwave for Electron Spin

The control of electron spin is performed by transverse AC magnetic field in microwave range.

$$\mathcal{H}_{\mathrm{MW}} = \sqrt{2}\Omega_{\mathrm{MW}}(t)\cos(\omega_{\mathrm{MW}}t + \phi_{\mathrm{MW}}(t))S_x$$

where Ω_{MW} is MW Rabi Amplitude which can be time-series value. ω_{MW} is set to resonance frequency of NV electron spin. Here, we implement π-pulse via MW operation.

NV Center System

Available Control Sources

RF for Nuclear Spin

It is possible to make a coherent drive between nuclear spin states in radio-frequency range of AC magnetic field.

$$\mathcal{H}_{\mathrm{RF}} = 2\Omega_{\mathrm{RF}}(t)\cos(\omega_{\mathrm{RF}}t + \phi_{\mathrm{RF}}(t))I_x$$

where Ω_{RF} is RF Rabi Amplitude which can be time series value. The driving field phase can also be changed during gate operation.

Note: RF pulses cannot perform selective nuclear spin control

Machine Learning Approach

Goal: Obtain a GHZ state after time evolution by selecting optimal parameters

Full Hamiltonian (with CPMG-based MW and RF)

(Electron spin in rotating frame, others are in lab frame)

$$\mathcal{H} = \underbrace{\sum_{i=1}^{N} \gamma_c B_z I_z^i + A_{\parallel}^i S_z I_z^i + A_{\perp} S_z I_x^i}_{\text{Drift Hamiltonian}} + \overbrace{\underbrace{\frac{1}{\sqrt{2}} \Omega_{\text{MW}}(t) S_x}_{\text{Operation on Electron Spin}} + \underbrace{\sum_{j=1}^{N} \left[\Omega_{\text{RF}}^j(t) \cos(\omega_{\text{RF}}^j t + \phi_{\text{RF}}^j(t)) \right] I_x^i}_{\text{Operation on } ^{13}\text{C spins}}}^{\text{time-dependent and controlable Hamiltonian}}$$

Cost Function

$$f = |\langle \text{GHZ}|U(t, t_0)|\psi_0\rangle|^2 = |\langle \text{GHZ}||\psi_f\rangle|^2$$

Cost function is the fidelity of the final state where $U(t, t_0) = \mathcal{T}\left[\exp\left(-i \int_{t_0}^{t} dt' \mathcal{H}(t')\right)\right]$.

Why Machine Learning Approach Fails

Issues:

1. GRAPE algorithm unavailable:

DDrf sequences for implementing gates on microsecond to millisecond timescales, while the control interval for optimization is on the order of nanoseconds. This creates an enormous optimization space with an impractical number of parameters to optimize.

2. Unknown optimal time: Even when employing known DDrf pulse sequences, determining the optimal total evolution time remains unclear. Since the number of π-pulses must be discrete (one cannot apply a fractional number of pulses), the total evolution time must be selected explicitly.

3. Gradient descent cannot consider realistic conditions: Gradient descent methods are designed to exploit the cost function—finding parameters that minimize the objective without inherent awareness of physical constraints.

Issue 3: Realistic Constraints

RF Amplitude (Ω_{RF}) Constraints

- **Theoretical:** Large amplitude causes unwanted detuning to other nuclear spins

- **Experimental:** Large amplitude can burn the wire

- **Problem:** Enforcing this condition in learning is not straightforward

Attempted Solutions

Q: Qutip's solver can be integrated with the scipy solver. What about adding bounds? Why not use a constrained (bounded) solver?

A: I tried the L-BFGS-B optimizer, but it failed, took too long, and ultimately did not converge.

Q: What about using a regularization method?

$$f_r(\Omega_{\mathrm{RF}}) = \Omega_{\mathrm{RF}}^2, \quad f \to f + \lambda f_r$$

A: `scipy.optimizer` failed to converge. Cannot debug the builtin optimizer.

Strategy

New Approach:

1. **Analytical analysis first** (Jiwon contributed significantly!)

   - Find good parameters beforehand

   - Use ML for fine-tuning only

2. **Better optimizer needed**

   - Use PyTorch with the Adam optimizer

   - Better convergence (personally experienced with GRAPE algorithm)

3. **New Schrödinger Equation Solver**

   - Implemented in PyTorch

   - Enables gradient-based optimization with better convergence

Solving Differential Equations

Time-Dependent Schrödinger Equation

$$\frac{\partial}{\partial t}|\psi\rangle = -i\mathcal{H}(t)|\psi\rangle$$

Naive Approach (Euler Method)

Consider a very simple model:

$$\mathcal{H} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

and initial state $|\psi\rangle = |0\rangle$. We can do something like this:

$$|\psi(t+dt)\rangle = |\psi(t)\rangle - idt\mathcal{H}|\psi(t)\rangle$$

(General idea of the Euler method)

Why Euler Method Fails

Norm Preserving Issue

Rightside figure: $dt = 10^{-2}$. Dotted result comes from Euler method.

Error proportional to $O(dt^2)$.

Runge-Kutta Method

RK4 Example (4th Order)

The Runge-Kutta method improves upon Euler by evaluating the derivative at multiple intermediate points:

$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + \frac{dt}{2}, y_n + \frac{dt}{2}k_1)$$
$$k_3 = f(t_n + \frac{dt}{2}, y_n + \frac{dt}{2}k_2)$$
$$k_4 = f(t_n + dt, y_n + dt k_3)$$
$$y_{n+1} = y_n + \frac{dt}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

For the Schrödinger equation $\frac{d}{dt}|\psi\rangle = -iH(t)|\psi\rangle$, we have $f(t, \psi) = -iH(t)|\psi\rangle$

Uses a n-th order method for higher accuracy with error $O(dt^n)$.

(Explicit) Runge-Kutta Method

The generalization of explicit Runge--Kutta methods is given by

$$y_{n+1} = y_n + dt \sum_{i=1}^{s} b_i k_i,$$

where

$$k_1 = f(t_n, y_n),$$
$$k_2 = f(t_n + c_2 dt, y_n + (a_{21} k_1) dt),$$
$$k_3 = f(t_n + c_3 dt, y_n + (a_{31} k_1 + a_{32} k_2) dt),$$
$$\vdots$$
$$k_s = f(t_n + c_s dt, y_n + (a_{s1} k_1 + a_{s2} k_2 + \cdots + a_{s,s-1} k_{s-1}) dt).$$

I ported code from `qutip/solver/integrator/qutip_integrator.py` . The details of the specific value of this method is shown here.

(Explicit) Runge-Kutta Method

I ran the same toy model simulation.

Key Features of Implementation

1. Interpolation

Given π pulse time, about few nano second, and its pulse shape(Gaussian), rapidly changing during π-pulse operation, time step must be much shorter than π-pulse duration. If we set `dt` in that way and save every vector or observable expectation value results, the amount of required memory would be large, considering potential larger simulation. So, to avoid this problem, I tried to take middle step during calculation between `dt`, larger compared to π-pulse time.

```python
def _interpolate_step(self, t: float) -> torch.Tensor:
    ...
    # Compute polynomial coefficients using Horner's method
    for i in range(self.rk_extra_step):
        for j in range(self.denseout_order - 1, -1, -1):
            self.b_factor[i] += self.bi[i, j] * tau
    # Interpolate: y = y_prev + dt * sum(b_factor[i] * k[i])
    ...
```

Key Features of Implementation

2. Norm Preserving Check

To ensure the norm of state vector to stay at 1, I checked this every (interpolated) time step.

```python
def _error(self, dt: float) -> float:
    ...
        # Compute error vector: dt * sum(e[i] * k[i])
        self._y_temp.zero_()
        self._accumulate(self._y_temp, self.e, dt, self.rk_step)

        self._norm_front = torch.norm(self._y_front).item()
        error_norm = torch.norm(self._y_temp).item()
        tol = self.atol + max(self._norm_prev, self._norm_front) * self.rtol

        return error_norm / tol
```

If error is not tolerable, the solver will use smaller `dt`.

Key Features of Implementation

3. Adaptive Time Step (Under development and Testing)

Checking how rapidly the Hamiltonian changes, determine interpolated time step adaptively. RF amplitude changes relatively slower than MW amplitude. That is, during π-pulse time, the solver use smaller interpolation step, otherwise, relatively larger interpolation step is acceptable.

```python
def _adaptive_dt_safe(self, t: float) -> float:
    ...
    # Apply bounds
    if self.max_step > 0:
        adaptive_step = min(self.max_step, adaptive_step)
    adaptive_step = max(self.min_step, adaptive_step)

    return adaptive_step
```

Original Purpose: Extending Coherence Time

CPMG was used to increase coherence time, but can be used for many purposes, including implementing conditional gates.

Hamiltonian

$$\mathcal{H} = \underbrace{\gamma_c B_z I_z^i + A_{||}^i S_z I_z^i + A_\perp S_z I_x^i}_{\text{Drift Hamiltonian}} + \underbrace{\frac{1}{\sqrt{2}} \Omega_{\text{MW}} S_x}_{\pi\text{-pulse}}$$

$$\rightarrow \mathcal{H} = |0\rangle\langle 0| \otimes \omega_0 I_z + |-1\rangle\langle -1| \otimes \left(\omega_L I_z - A_{||} I_z - A_\perp I_x \right)$$

$$\rightarrow \mathcal{H} = |0\rangle\langle 0| \otimes H_0 + |-1\rangle\langle -1| \otimes H_1$$

That is, nuclear spin evolves by $e^{-iH_{0(1)}t}$ when electron spin lies on spin state $|0\rangle (|1\rangle)$.

Figure. Diagram for CPMG sequence/ CPMG is a repetition of $(\tau - \pi - 2\tau - \pi - \tau)$ for $\frac{N}{2}$ times. The figure is extracted from `Phys. Rev. X 15, 021011`.

CPMG Sequence

Unitary Operations

- Initial electron spin state 0(1): $e^{-i\phi \hat{I} \cdot \hat{\sigma}^i}$

  - if NV spin is 0-state: $e^{-i\phi \hat{I} \cdot \hat{\sigma}^0} = e^{-iH_0\tau} e^{-iH_1 2\tau} e^{-iH_0\tau}$

  - if NV spin is 1-state: $e^{-i\phi \hat{I} \cdot \hat{\sigma}^1} = e^{-iH_1\tau} e^{-iH_0 2\tau} e^{-iH_1\tau}$

- When vectors σ^0 and σ^1 are anti-parallel $\rightarrow$ conditional gate

- Directions are generally parallel, but at a specific τ they become anti-parallel

Note: ϕ is equal whatever initial state is 0 or 1 in CPMG train.

Finding τ Analytically(?)

To enable conditional Gate via repeatation of CPMG train, we should choose τ such that $\hat{\sigma^0} \cdot \hat{\sigma^1} = -1$.

τ can be found analytically under strong magnetic field. [Phys. Rev. Lett. 109, 137602]

$$\tau \simeq \frac{(2k-1)\pi}{2\gamma_c B_z + A_{||}}$$

Then, the number of repeatation number of CPMG train is chosen meticulously, to acheive $N\phi = \frac{\pi}{2}$.

Conditional Operation

At a certain τ, we observe a conditional operation. Red(Blue) arrow denotes rotating axis where initial eletron spin state was $|0\rangle$ ($|1\rangle$).

Multi-Qubit System

In a multi-qubit system, one nuclear spin responds to the CPMG sequence.

$$A_{\parallel} = [-0.130, -0.210, 0.045] \qquad [\text{MHz}]$$
$$A_{\perp} = [0.050, 0.003, 0.025] \qquad [\text{MHz}]$$

Multi-Qubit CPMG

[TODO] DDrf: RF Pulse Control

RF pulses can be employed to add additional active operations on nuclear spins

Conditional Operation (CRX Gate)

With meticulous RF phase updates, we can derive a conditional operation. Here, for DDRF gates, the interpulse delays do not need to follow the dynamics of the target nuclear spin: Direct spin-state selective radio-frequency driving with tailored phase updating enables a conditional rotation of the nuclear spin.

rightside

TODO

1. **Analytical study:** Find better parameters

2. **Polish code:**

   - Memory management

   - Find better hyperparameters for faster simulation

   - Reduce simulation time to ~1min (for learning purposes)

3. **Run learning** after obtaining good parameters

4. **(Later) Develop density matrix time-evo solver in PyTorch**

   - NV/nuclear spins are not fully initialized to pure states

   - Need to consider mixed states → density matrix approach