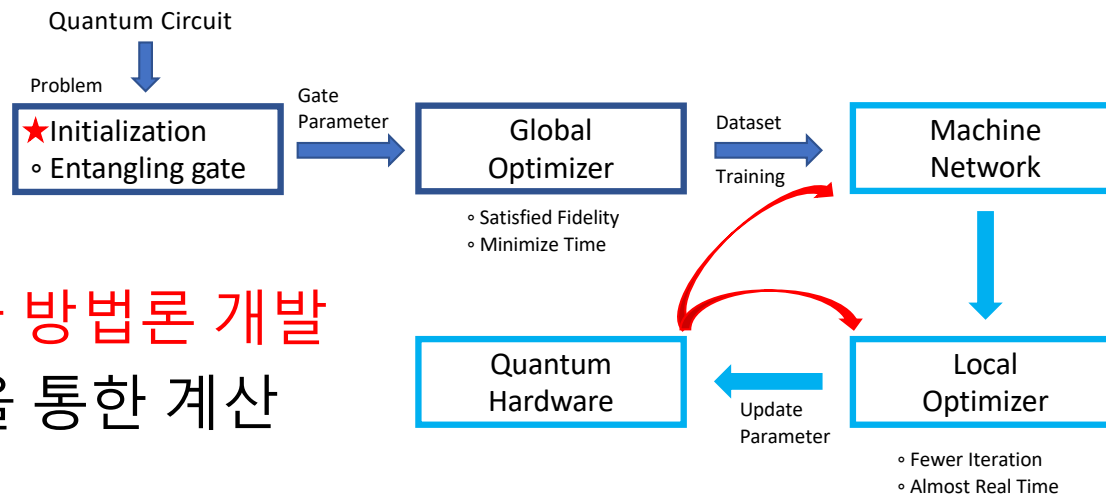


Time- Optimal NV Spin Control

- 양자정보연구단 인턴 하규원

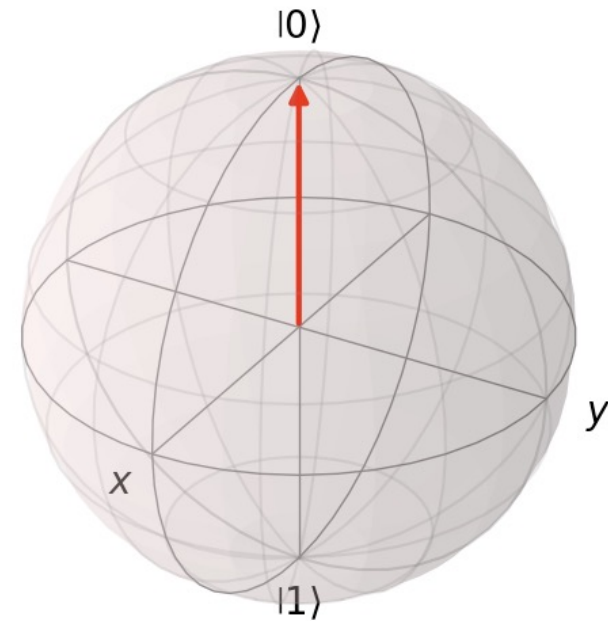
목표

- Spin control에 대한 방법론 개발
- Machine Learning을 통한 계산 속도 개선
- 실제 실험 셋업에 적용



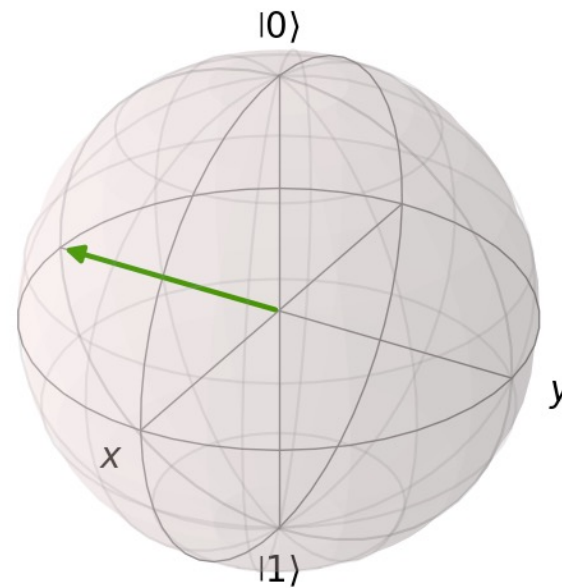
Spin control 방법론적 개발 (Single NV spin)

- $|0\rangle$ state로 초기화 되어있는 NV spin을 pulse를 이용하여 최단시간 내에 원하는 상태로 보내는 코드를 개발한다.
- Single qubit 대해서 성공한 후, $^{12}\text{Carbon}$ -Nuclear spin에 적용한다.



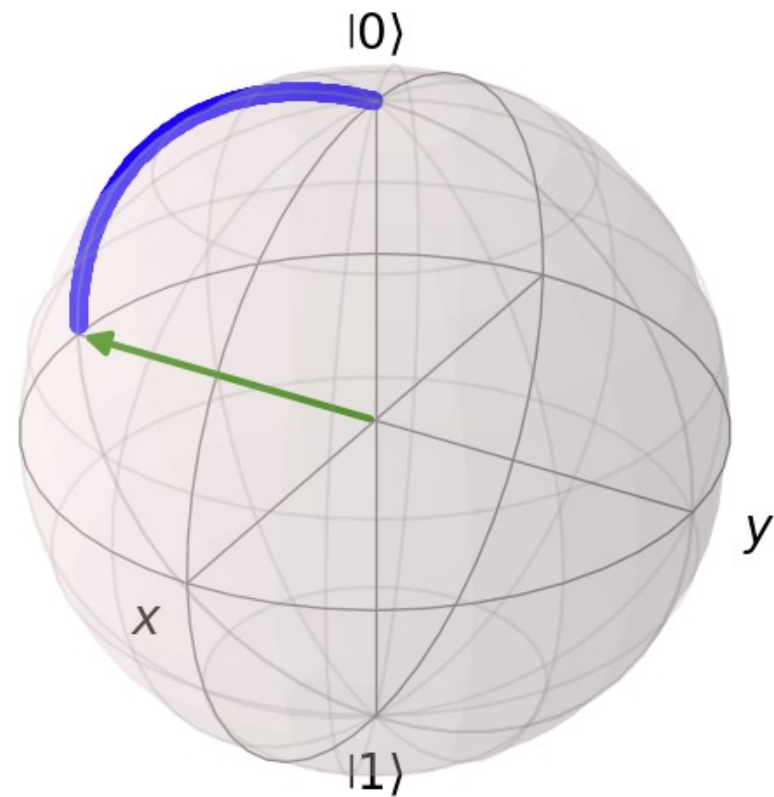
Spin control 방법론적 개발 (Single NV spin)

- 만약, spin을 중첩상태로 보내고 싶다면, +x축 pulse를 가하면, 최단경로로 이동할 수 있을 것이다.



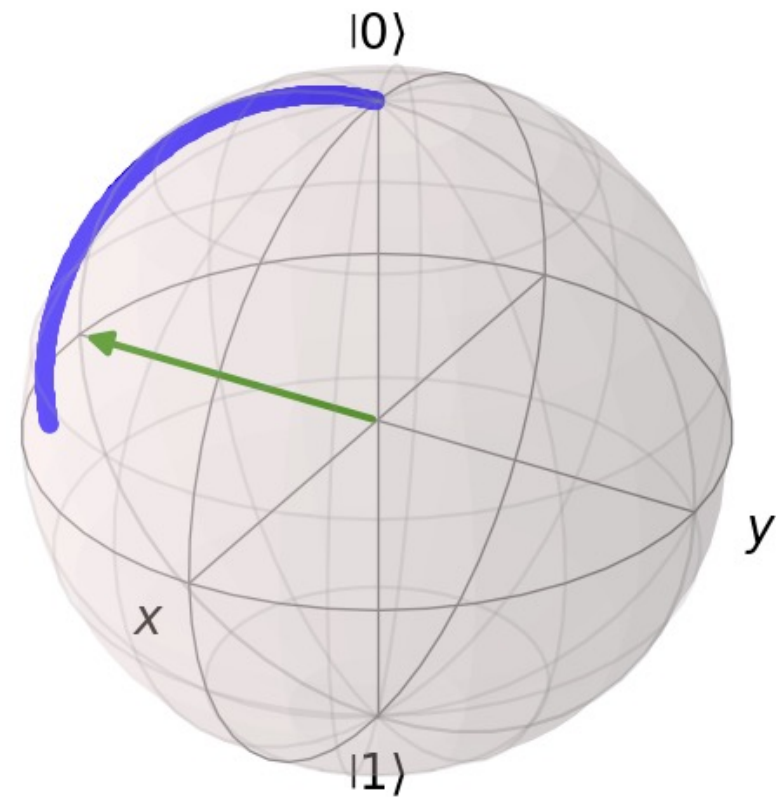
Spin control 방법론적 개발 (Single NV spin)

- +x 방향의 pulse를 $\pi/2$ 만큼 회전하게
짜준다면 이와 같은 경로로 이동하게
됩니다.



Spin control 방법론적 개발 (Single NV spin)

- 하지만, Detuning에 의해 +z 회전이 발생하는 시스템이라면, 그림과 같이 원하는 상태로 보낼 수 없게 된다.





Detunning

- $H(t) = H_d + H_c(t)$
- $H_d = d_0 \cdot s_z$ (*detunning term*)
- $H_c(t) = v_1 \cdot [c_1(t) \cdot s_x + c_2(t) \cdot s_y]$ (*control term*)

$$d_0 = 0.15 \text{ rad/ms}$$

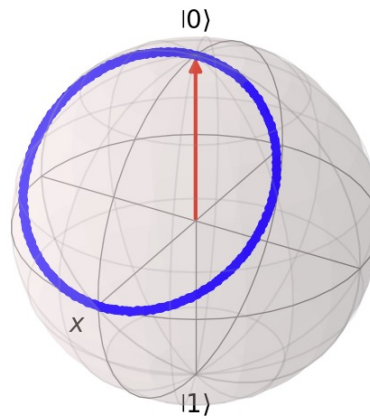
$$v_1 = 0.02 \text{ rad/ms}$$

$c(t)$ 는 시간에 따른 pulse의 방향을 선택하는 step function
->각 step마다 -1,0,+1 값 중에서 하나를 갖는다.

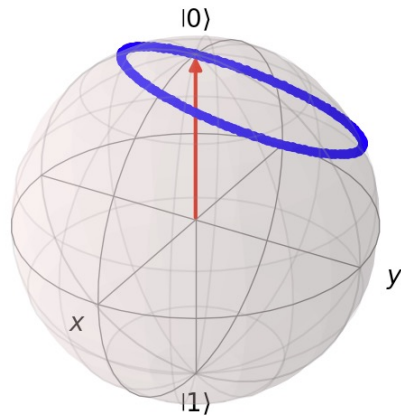


Detuning term을 넣어서 계산하는 이유

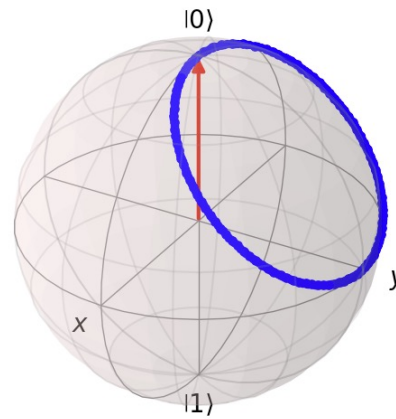
- 실제 실험 셋업에서 Spin control은 다양한 노이즈가 존재.
→ 노이즈 환경에서 원하는 게이트를 구현.
 - ^{13}C Carbon Nuclear Spin 컨트롤에서는 s_z 와 s_x term 존재.
→ 이를 단순화한 문제에서, 테스트 하기 위함.
-



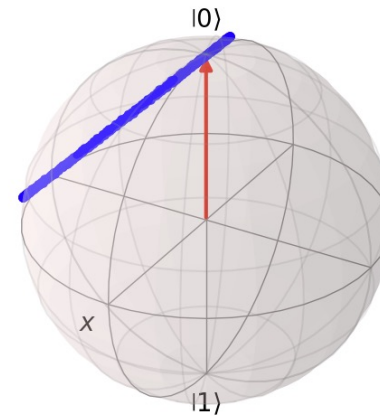
+x rotation



-x rotation



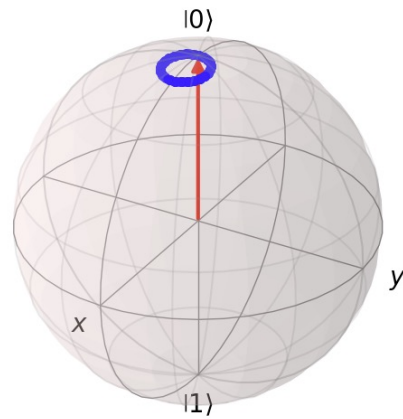
+y rotation



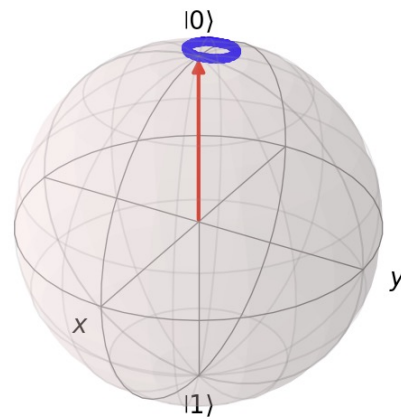
-y rotation

Hamiltonian에
의한 회전축

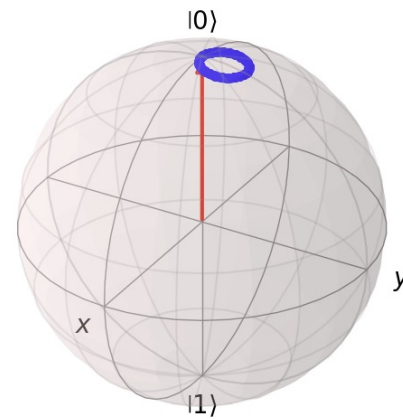
- $d_0 = 0.02 \text{ rad}/\mu\text{s}$
- $v_1 = 0.02 \text{ rad}/\mu\text{s}$



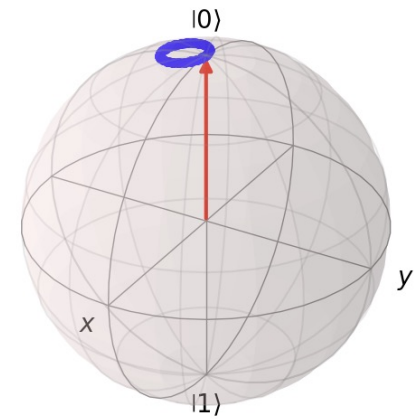
+x rotation



-x rotation



+y rotation



-y rotation

Hamiltonian에
의한 회전축

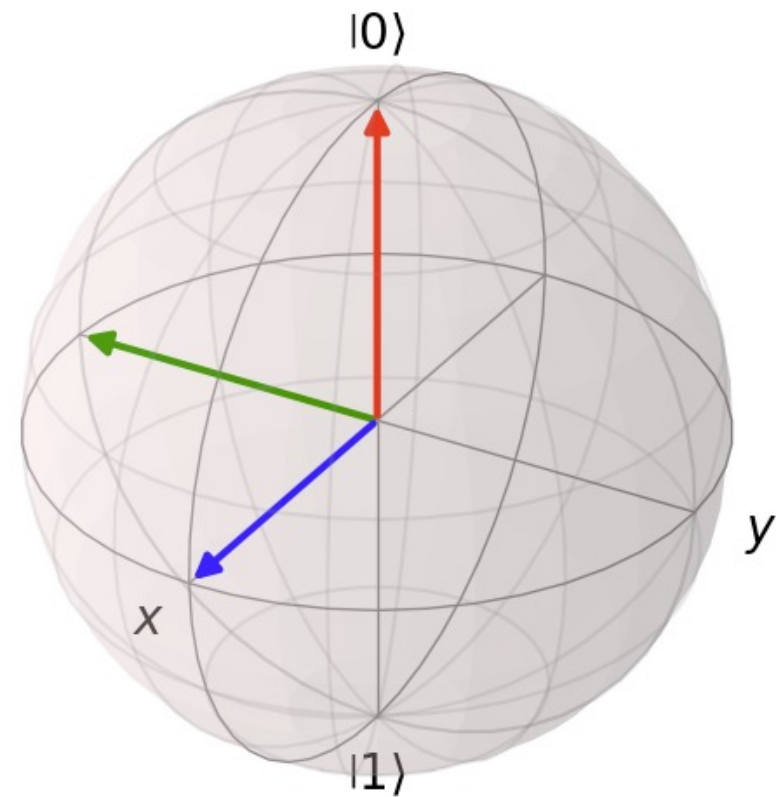
- $d_0 = 0.15 \text{ rad}/\mu\text{s}$
- $v_1 = 0.02 \text{ rad}/\mu\text{s}$

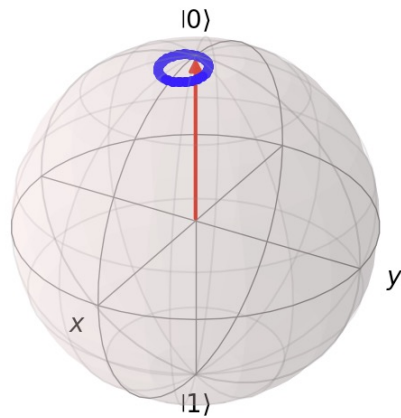
최적화 방법

- 시간에 따라서 Hamiltonian이 변화하므로, 구간을 이산적으로 나눠서 순간마다 최적의 pulse를 찾는다.
 - Global optimizer를 사용하여, 적절한 시간구간 dt 를 찾는다.
 - Fidelity를 만족하면 탐색 중지
 - Fidelity = $F(\rho, \sigma) = \left(\text{tr} \sqrt{\sqrt{\rho}\sigma\sqrt{\rho}} \right)^2$
-

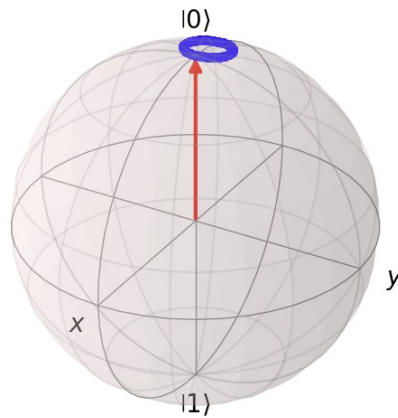
최적화 방법

- 빨간색 : initial state
- 초록색 : target state
- 파란색 : ideal rotation axis
- 빨간색 벡터성분과 초록색 벡터성분의 외적을 통해 ideal rotation axis를 찾을 수 있다.

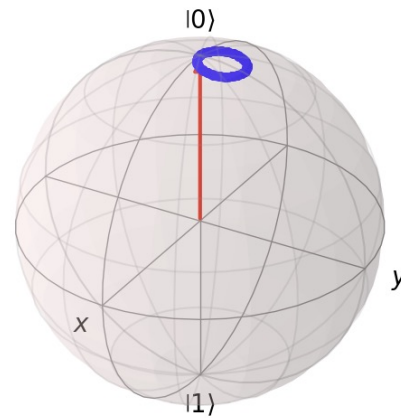




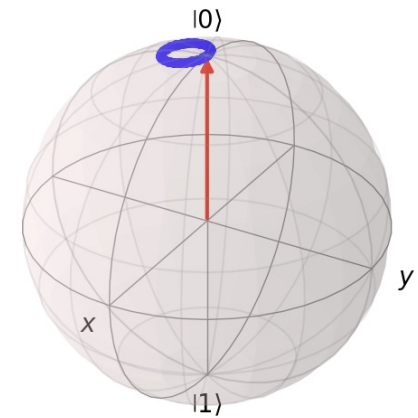
+x rotation



-x rotation



+y rotation



-y rotation

최적화 방법

- Ideal rotation axis와 가장 가까운 축을 갖는 pulse를 선택한다.
- 가장 가까운 축은 파란색 축과 rotation축 사이의 내적값을 통해 판단한다. 내적값이 큰 축을선택.
- 현재 상태에서는 +x rotation pulse 선택

최적화 방법

- 첫번째 step에서의 pulse를 선택한 후, 두번째 step부터는 이전step내적값과 현재 step 내적값의 변화량을 비교한다.
- 변화량이 큰 축을 선택하게 된다.

```
if step > 1 :  
    for j in range(4) :  
        delta[j] = inner_product[j] - past_inner_product[j]  
    # 다음 계산을 위한 내적값 저장  
    past_inner_product = inner_product.copy()  
  
    max_delta_value = max(delta)  
    max_delta_indices = [j for j, v in enumerate(delta) if v==max_delta_value]
```

최적화 방법

- 만약 변화량이 같은 축이 존재한다면, 그 축들 중에서 현재 step의 내적값이 더 큰 축을 선택하게 된다.

```
else :  
    for i in range(4):  
        if i not in max_delta_indices :  
            inner_product[i] = -3  
    max_value = max(inner_product)  
    max_indices = [i for i, v in enumerate(inner_product) if v==max_value]  
    choice = random.choice(max_indices)
```



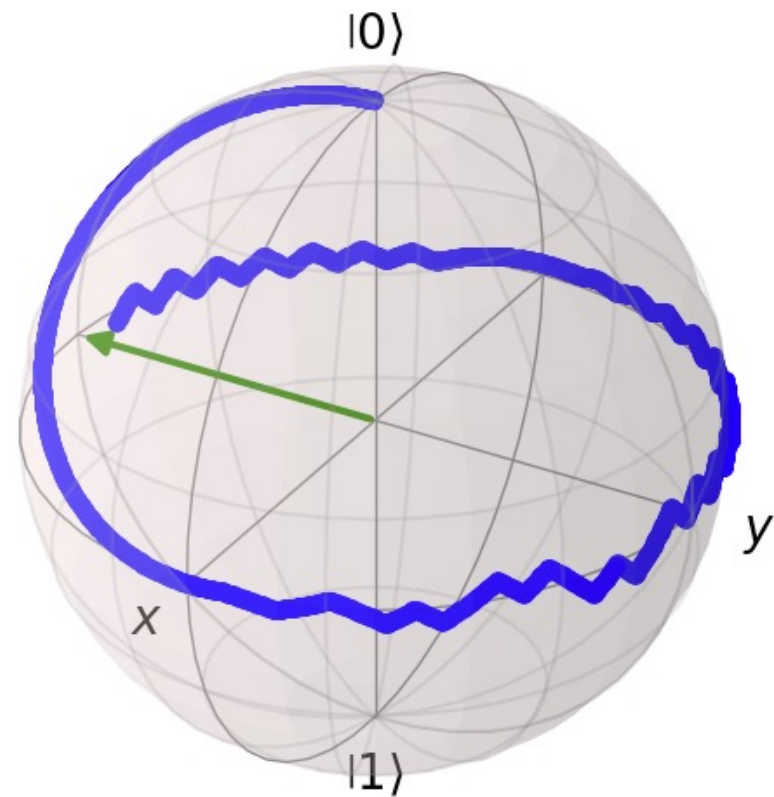
최적화 방법

- 이러한 constraints를 통해 각 시구간(dt)에서의 적절한 pulse를 찾게된다.
- 시구간dt는 Global Optimizer인 SHGO Optimizer를 사용하여 탐색한다.

```
bounds = [(1,15)]  
rlt = optimize.shgo(make_combination,bounds=bounds,_iters=6, options={'xtol':1e-15,'ftol':1e-15})  
output.append(['CASE'+str(t+1),len(trace[1]),target_theta,target_phi,trace[2],trace[1],rlt['fun'], rlt['nfev']])
```

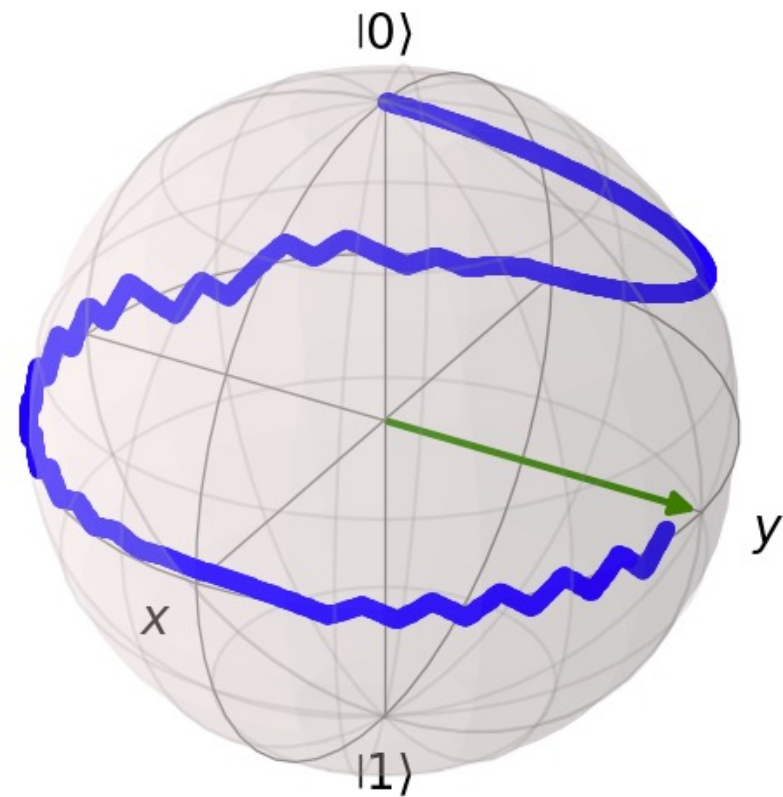
시뮬레이션 결과

- $d_0 = 0.02 \text{ rad}/\mu\text{s}$
- $v_1 = 0.02 \text{ rad}/\mu\text{s}$
- Number of step = 90
- $dt = 1.88 \mu\text{s}$
- Total time = $169.2 \mu\text{s}$
- Fidelity = 0.99900



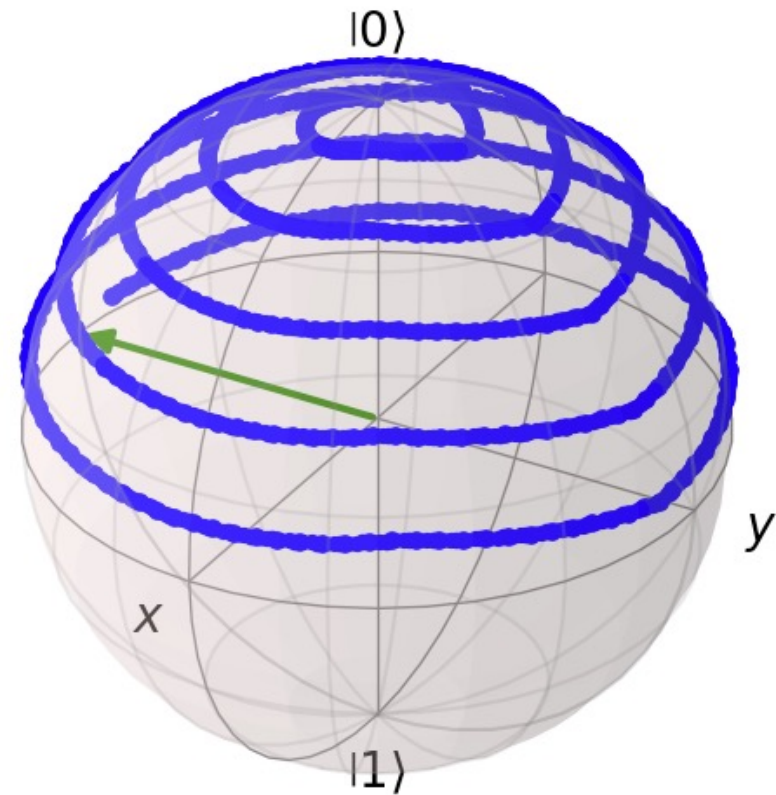
시뮬레이션 결과

- $d_0 = 0.02 \text{ rad}/\mu\text{s}$
- $v_1 = 0.02 \text{ rad}/\mu\text{s}$
- Number of step = 73
- $dt = 2.31 \mu\text{s}$
- Total time = $168.63 \mu\text{s}$
- Fidelity = 0.99852



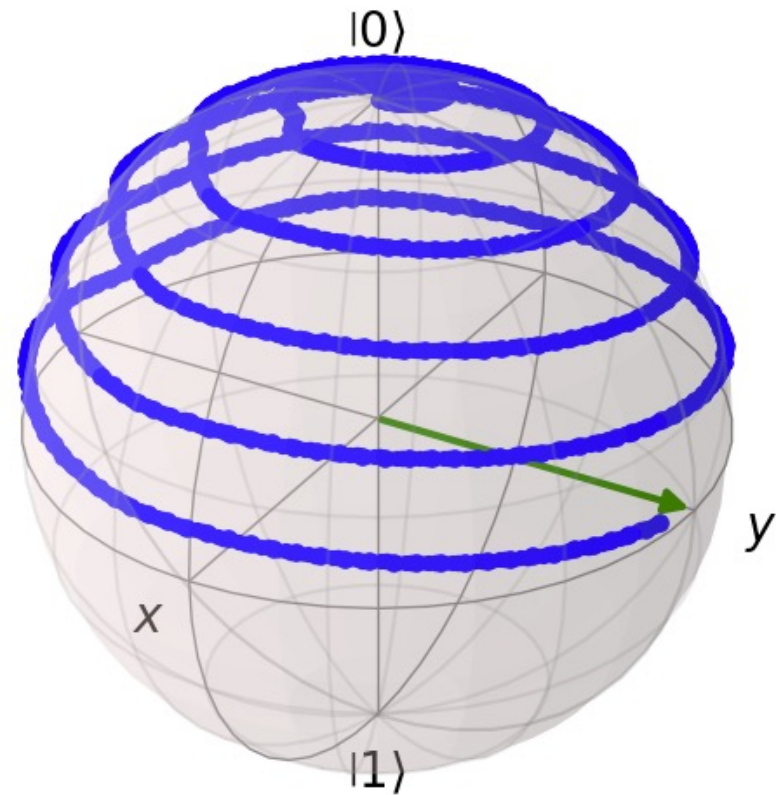
시뮬레이션 결과

- $d_0 = 0.15 \text{ rad}/\mu\text{s}$
- $v_1 = 0.02 \text{ rad}/\mu\text{s}$
- Number of step = 42
- $dt = 2.63 \mu\text{s}$
- Total time = $110.46 \mu\text{s}$
- Fidelity = 0.99963



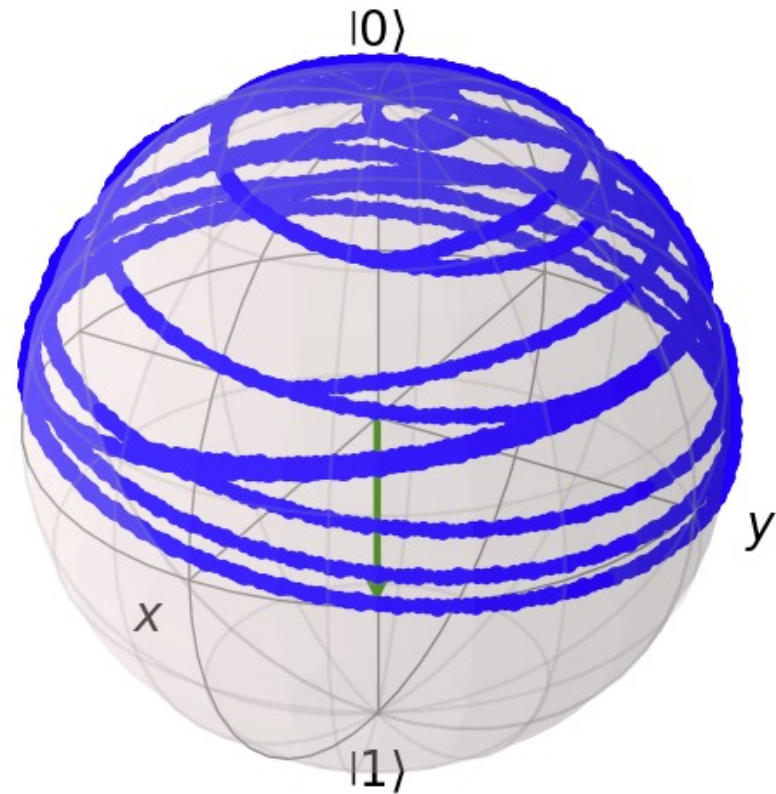
시뮬레이션 결과

- $d_0 = 0.15 \text{ rad}/\mu\text{s}$
- $v_1 = 0.02 \text{ rad}/\mu\text{s}$
- Number of step = 42
- $dt = 2.63 \mu\text{s}$
- Total time = $110.46 \mu\text{s}$
- Fidelity = 0.99868



개선해야할 점

1. 원하는 target state로 가지 못하고 시간을 소모하는 경우가 있음
 - 현재 사용하고 있는 constraints보다 더 엄격하게 하는 것은 문제를 푸는데에 유연성을 주기 힘들다.
 - 최적화 알고리즘에서 현재 dt 만을 최적화 하고 있는데, choice도 최적화 변수로 넣어서 유연하게 문제를 풀 수 있는 방안 찾기



개선해야할 점


2. 현재 cost function은 오직 state fidelity만 고려하고 있음

$$\text{cost} = 1 - F(\rho, \sigma)$$

→ 시간에 따른 penalty 부여

$$\text{cost} = 1 - F(\rho, \sigma) + \kappa \int_0^T dt$$

→ 시간에 대한 가중치 κ 의 적절한 값을 찾아야한다.



향후 목표

1. Time-optimal한 데이터(dt, pulse 조합)들을 뽑아서, ML로 학습시킨다.
2. ML을 통해 원하는 state로 보내기 위한 dt와 pulse 조합을 빠르게 추정한다.
3. 이를 실제 실험 셋업에 적용시킨다.

최종 목표 : 이러한 과정을 ^{13}C Carbon Nuclear Spin에도 적용한다.

감사합니다