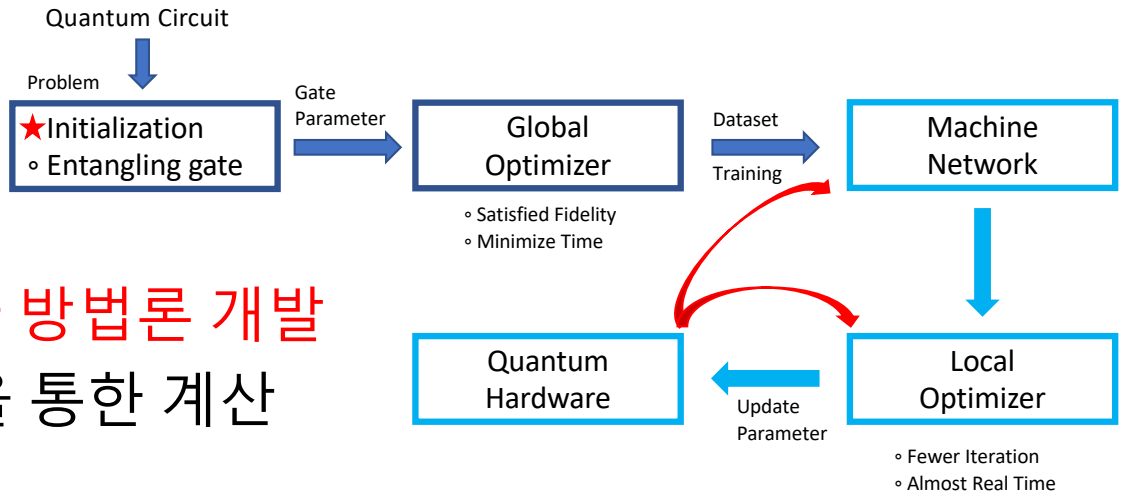
Time-Optimal NV Spin Control

- 양자정보연구단 인턴 하규원

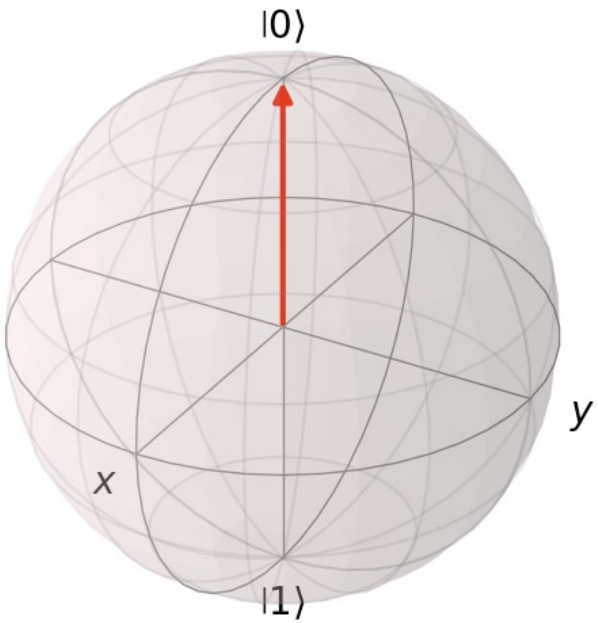
목표



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

Spin control 방법론적 개발 (Single NV spin)

- |0> state로 초기화 되어있는 NV spin을 pulse를 이용하여 최단시간 내에 원하는 상태로 보내는 코드를 개발한다.
- Single qubit 대해서 성공한 후, 13Carbon-Nuclear spin에 적용한다.



Detunning

- $H(\mathrm{t}) = H_{\mathrm{d}} + H_{\mathrm{c}}(t)$

- $H_{\mathrm{d}} = d_0 \cdot s_z \ (detunning\ term)$
- $H_{\mathrm{c}}(t) = v_1 \cdot \left[c_1(t) \cdot s_x + c_2(t) \cdot s_y\right] (control\ term)$

d_0 = 0.15 rad/ms

v_1 = 0.02 rad/ms

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

Detunning term을 넣어서 계산하는 이유

- 실제 실험 셋업에서 Spin control은 다양한 노이즈가 존재.

→노이즈 환경에서 원하는 게이트를 구현.

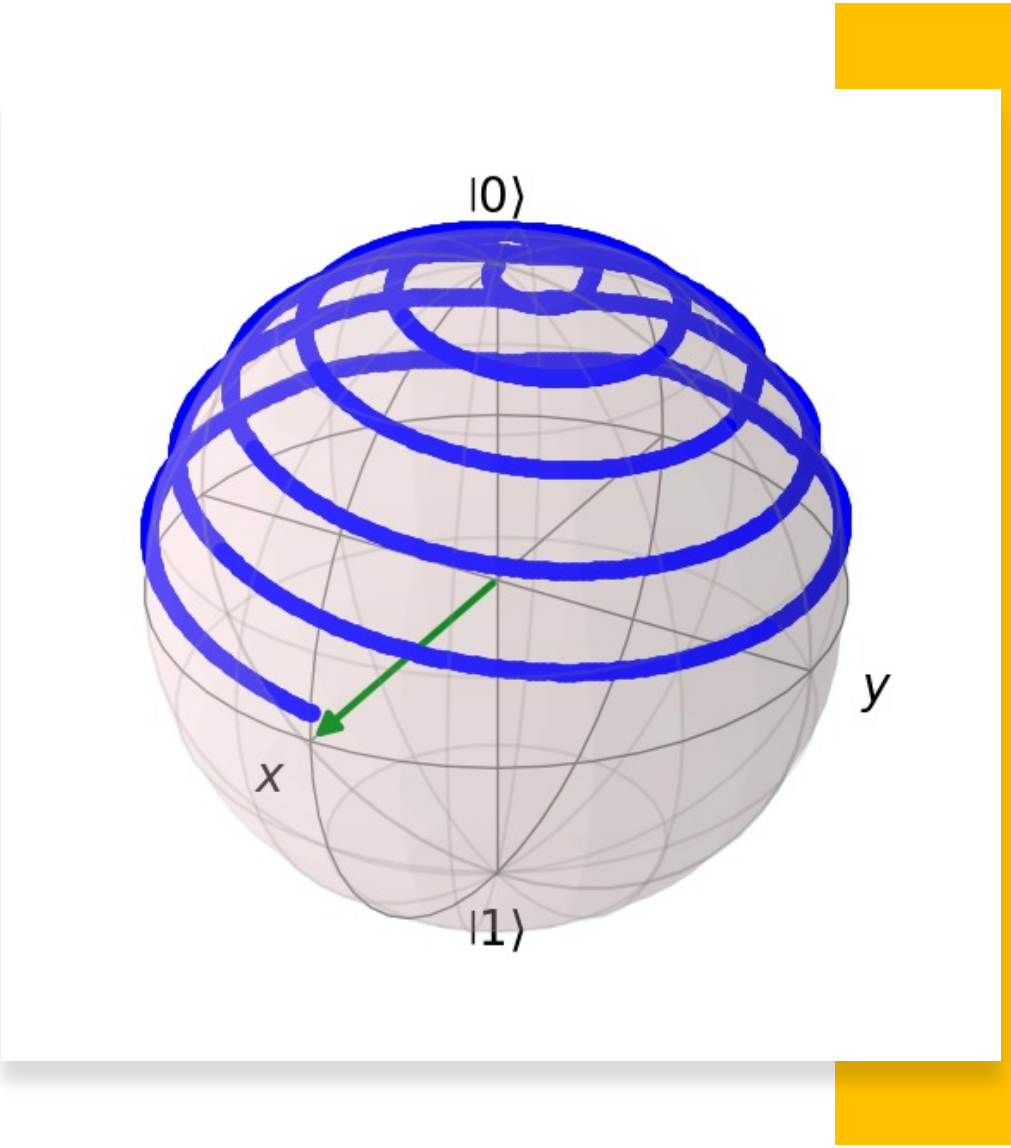
- 13Carbon Nuclear Spin 컨트롤에서는 s_z 와 s_x term존재.

→ 이를 단순화한 문제에서, 테스트 하기 위함.

기존 방식

- Current vector와 Target vector의 외적값 계산을 통한 회전축 선택

- d_0 = 0.30 rad/μs
- v_1 = 0.04 rad/μs
- Number of step = 63
- dt = 1.76 μs
- Total time = 110.88 μs
- Fidelity = 0.99807



개선된 방식

- *Random Search*
- A* search Algorithm(*Quantum Speed Limit*)

1. Random Search

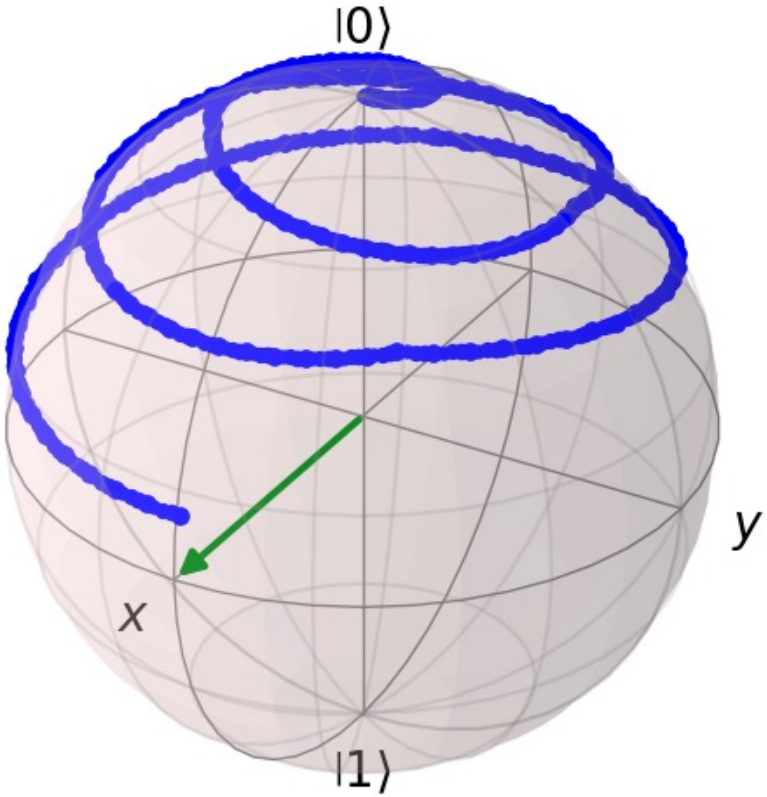
- 각 step마다 5개의 회전축 중에서, 랜덤으로 선택하는 방식
- Fidelity 0.99를 만족하면 pulse조합 반환
- Iteration을 반복하면서, 기존 pulse보다 짧은 시간에 도달하는 케이스만 반환

```
choice = random.randint(0,4)
instant_unitary = unitary(dt, choice)
irho_current = instant_unitary @ irho_current @ instant_unitary.conj().T

F = 1 - state_fidelity(irho_target,irho_current)
```

1. Random Search

- d_0 = 0.30 rad/μs
- v_1 = 0.04 rad/μs
- Number of step = 7
- dt = 9.5 μs
- Total time = 66.5 μs
- Fidelity = 0.99223
-



1. Random Search

- 장점
- 기존 방식에 비해 제약 조건이 아예 없기 때문에, 범용성이 크다.
- 연산량이 적기 때문에, 많은 반복을 통해 좋은 solution을 찾을 수 있다.

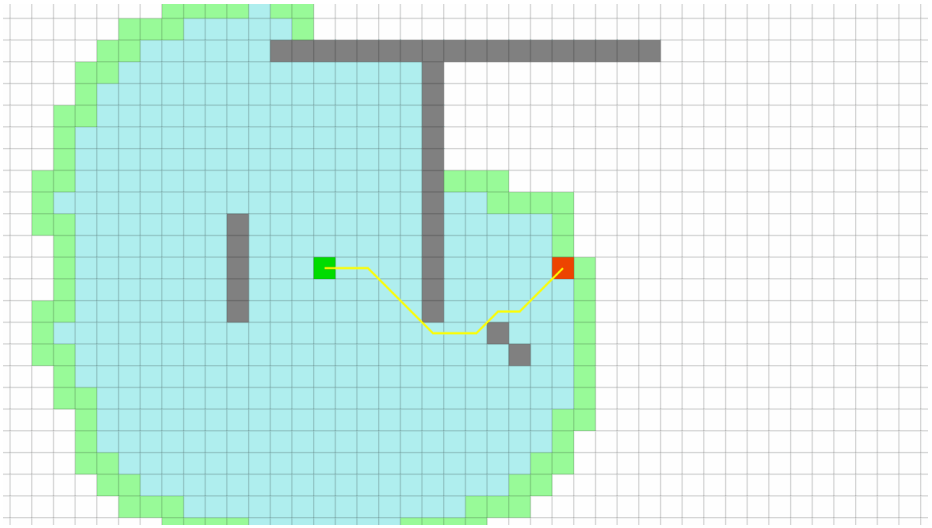
- 단점
- Dt가 작아지면, pulse조합이 필연적으로 길어지는데, 이로 인해 연산량이 기하급수적으로 증가한다.
- Initial state에서 target state가 멀수록, 연산량이 증가한다.

2. A*Search Algorithm

- 다익스트라 방식은 3차원의 Bloch Sphere에서 사용하기에 연산시간이 너무 소모됨.

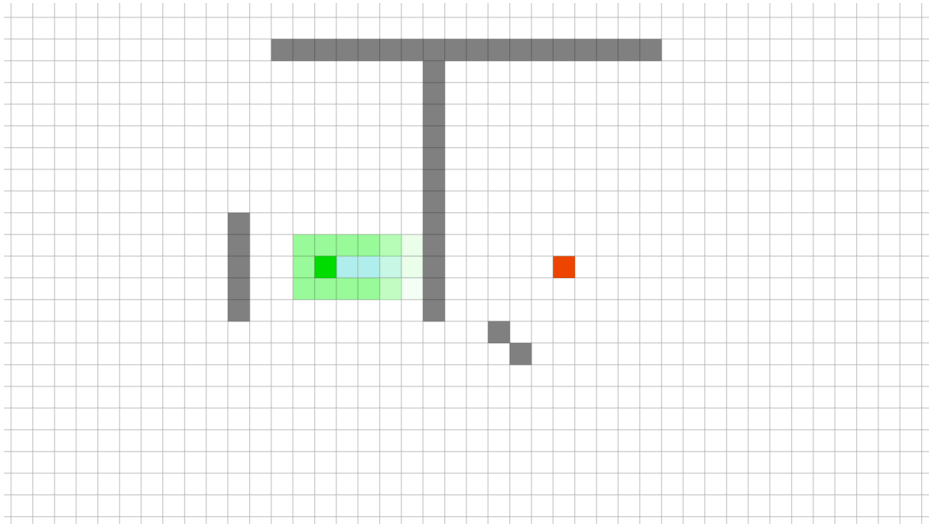
- Heuristic이라는 예상 시간 가중치를 사용한 A* Search Algorithm을 사용하는 것이 합리적.

2. A*Search Algorithm



다익스트라
알고리즘을 통한
길찾기

2. A*Search Algorithm



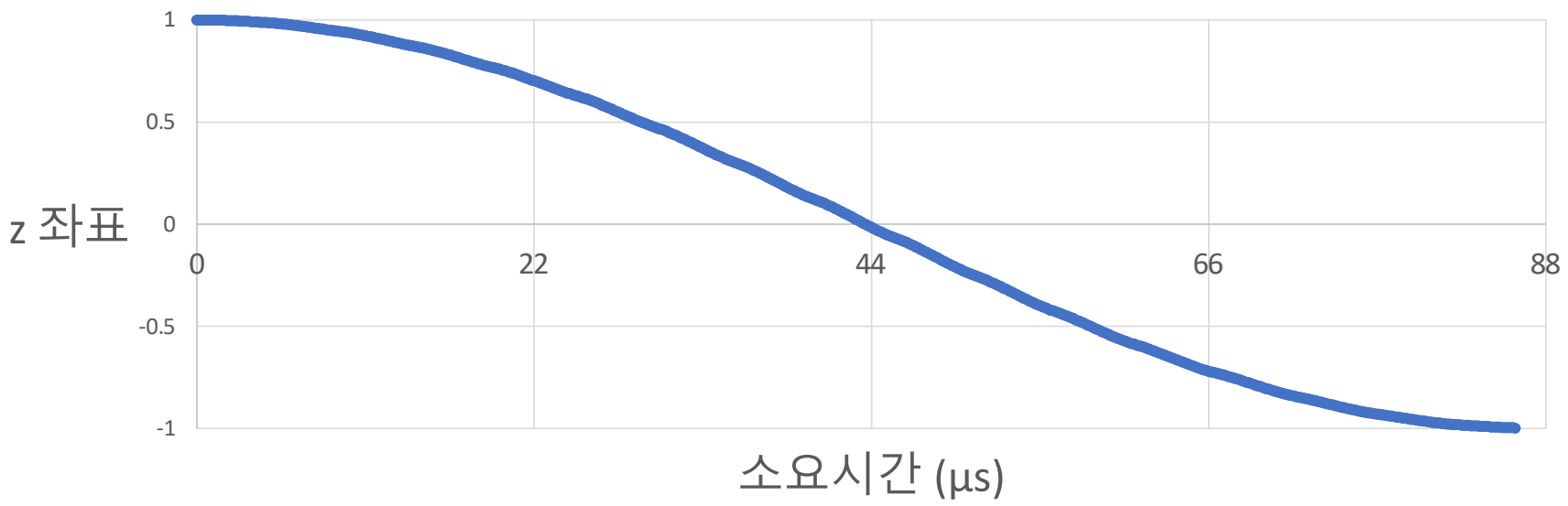
A* Search
알고리즘을 통한
길찾기

Quantum Speed Limit(QSL)

- |0> state에서 |1> state로 보내는 데에 걸리는 최단 시간을 계산함으로써, z좌표 위치에 따른 예상 최단시간을 계산할 수 있음

```
for i in range(1,5) :
    instant_U = unitary(dt,i)
    irho_temp = instant_U @ irho_current @ instant_U.conj().
    delta = current_z -  np.trace(irho_temp*sz).real
    temp_z.append(delta)
```

Quantum Speed Limit(QSL)



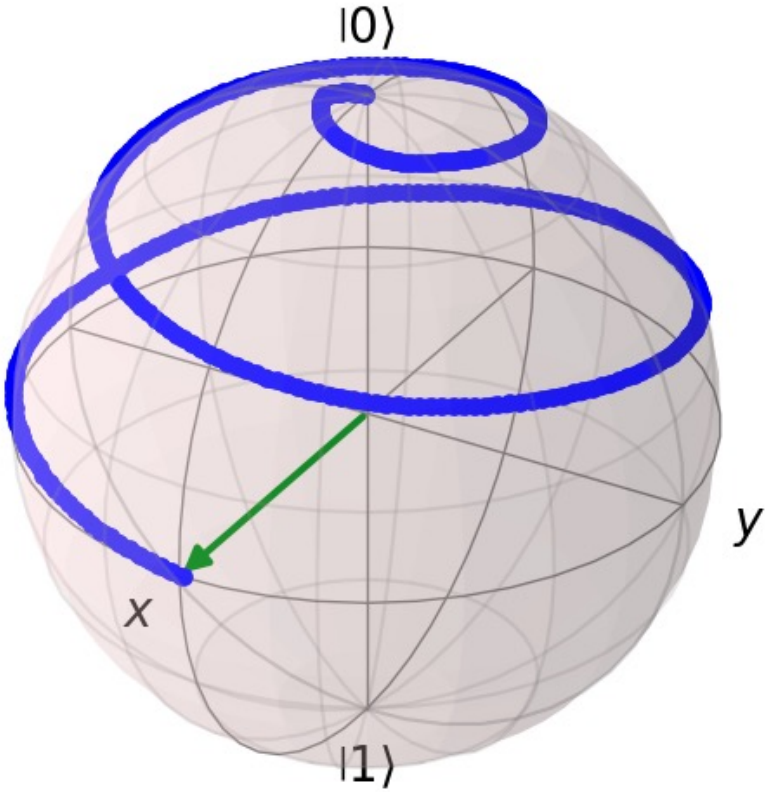
2. A*Search Algorithm

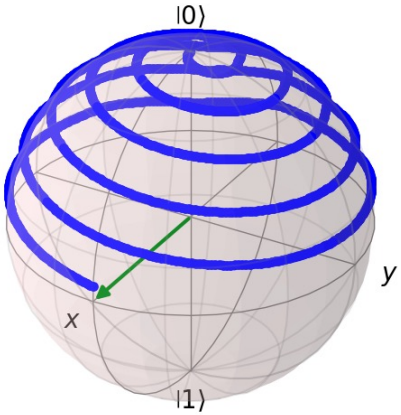
- 앞서 얻은 cos함수를 통해 z좌표에 대한 휴리스틱 계산가능
- 위상 정보에 대한 휴리스틱은 보정 값을 주어서 계산함.

```
# QSL을 통한, heuristic 추정
def heuristic(node,target):
    delta_z = abs((acos(target.z)-acos(node.z)) / 0.0365)
    phase = target.phase - node.phase+0.1726*theta
    if phase < 0 :
        phase+=2*pi
    phi = phase*0.1
    return delta_z +phi
```

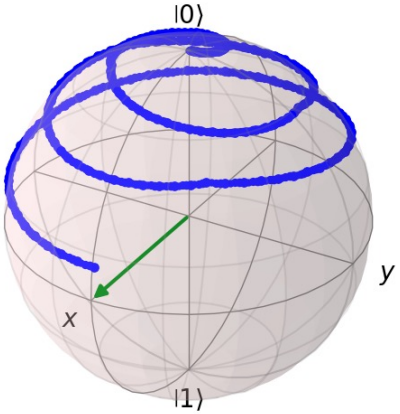
2. A*Search Algorithm

- d_0 = 0.30 rad/μs
- v_1 = 0.04 rad/μs
- Number of step = 16
- dt = 3 μs
- Total time = 48 μs
- Fidelity = 0.99996

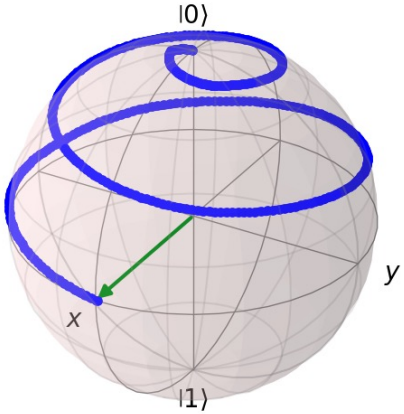




기존방식 (110.88 μs)



Random Search(66.5 μs)



A* Search(48 μs)

결과 비교

- A*알고리즘이 가장 optimal한 경로를 찾는다!

향후 목표

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

최종 목표 : 이러한 과정을 13Carbon Nuclear Spin에도 적용한다.

감사합니다