Reinforcement learning

IN DIFFERENT PHASES OF QUANTUM CONTROL

BY JASON BERGELT

Content

- Introduction
  - What do we want to achieve?
  - Why do we use reinforcement learning?

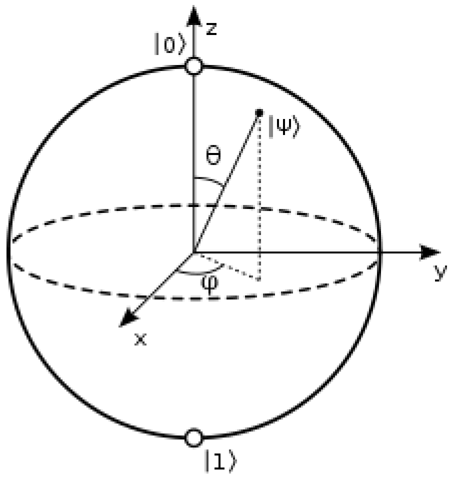
- Finite Markov Decision Processes
  - Formalism
  - Optimization
  - Exploration-exploitation dilemma

- Qubit controlling
  - Different phases of Quantum control

Introduction

What do we want to achieve?

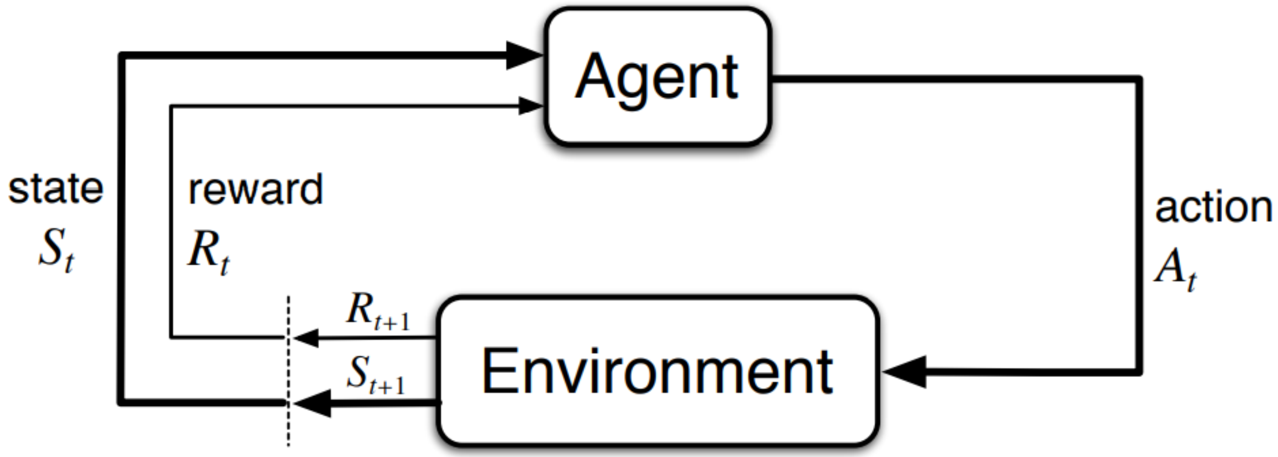
- *<u>Main goal:</u>*
  - Control quantum states in large quantum many-body systems
    - In our case: control single qubit
      - „Normal“ bit has two states 0 & 1
      - in contrast our qubit has states $|\Psi\rangle = a\,|0\rangle + b|1\rangle$
      - **Goal: prepare a specific target state**



- *<u>Challenges:</u>*
  - Lack of limited theoretical understanding
  - Complexity of simulating
    - experimental systems are uncontrollable
      - No finite-duration protocol to prepare the desire state
      - Adiabatic limit for non equilibrium systems often does not exist

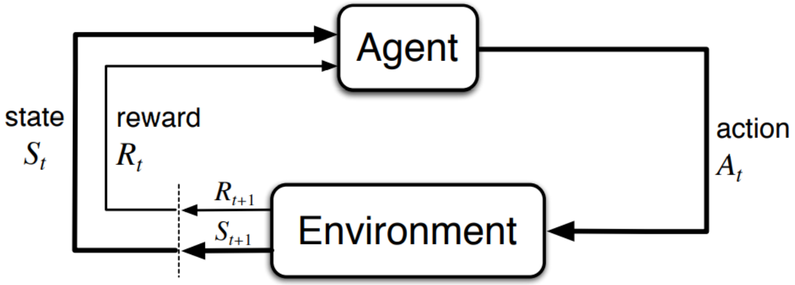
Why do we use reinforcement learning?

- Provides deep insight into nonequilibrium quantum dynamics
  - **without** knowledge of any model
- Discovers a stable suboptimal protocol
  - rivals optimal solutions in performance
    - **But: stable to local perturbation**
- Well suited to work with experimental data
  - Does not require the knowledge of the local gradient of control landscape
  - Advantage in controlling for difficult models *(difficulty due to disorder or dislocation)*



Finite Markov Decision Processes (MDP)

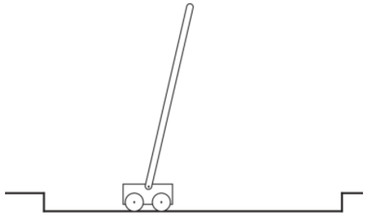
Agent vs Environment



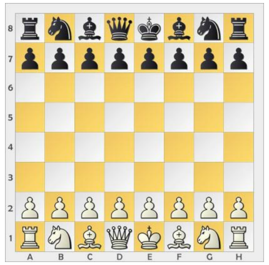
AGENT

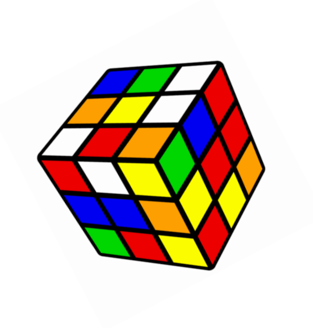
- the "learner"
  - can be anything that wants to learn

ENVIRONMENT



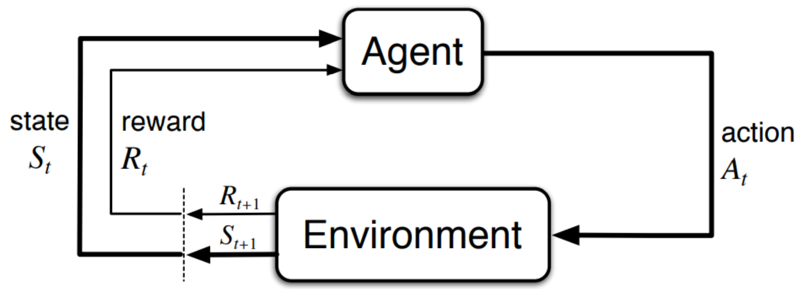
- anything the agent can't change
  - agent maybe knows everything about the environment
    - **But** still faces difficult reinforcement learning task
      - *Example: Rubik's Cube*





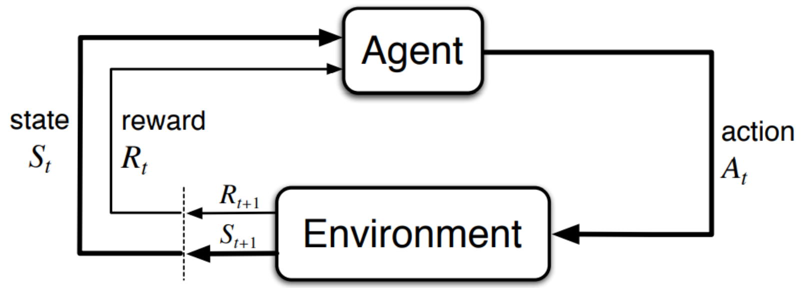
Agent-Environment boundary = limit of agent's absolute control

dynamics



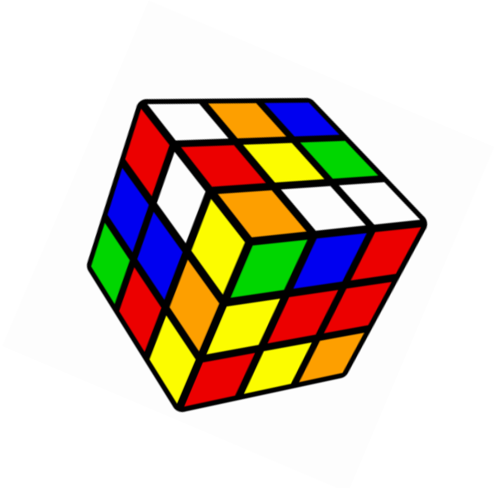
- set by the environment
- the dynamics $p: S \times R \times S \times A \to [0,1]$
  - $p(s', r | s, a) \coloneqq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$
    - ➢ $\sum_{s' \in S} \sum_{r \in R} p(s', r | s, a) = 1$ for all $s \in S, a \in A(s)$

- **later:**
  - Qubit $\xrightarrow{\text{Environment}}$ Schrödinger equation
    - ➢ Dynamics become deterministic
      - ➢ Probability = delta-distribution



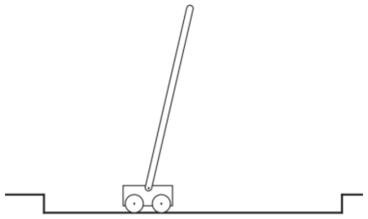
State space $\mathcal{S}$

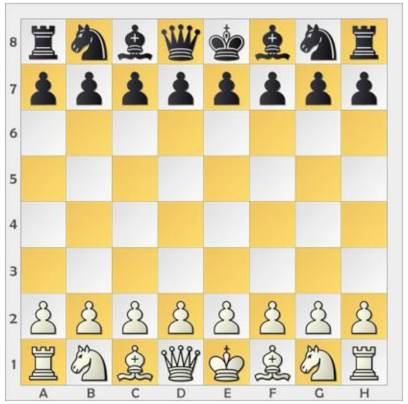
- State = "anything that might be useful information to the agent"



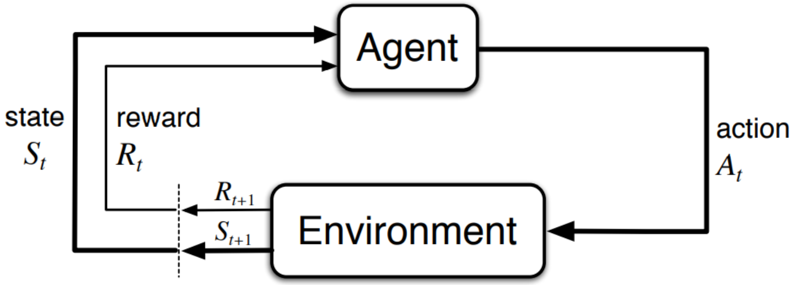
- includes information about the past agent-environment interaction

- At each time step t the agent chooses an action A_t based on state S_t
  - *so called: Markov property*
  - Received reward R_{t+1} and new state S_{t+1}
    - Dependent on preceding state and action
      - Via discrete probability distribution

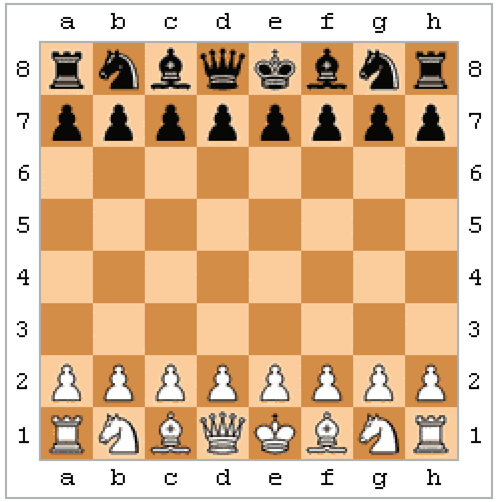


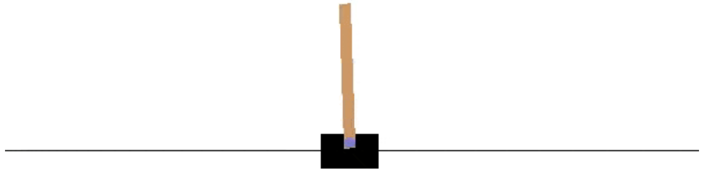


Action space $\mathcal{A}$

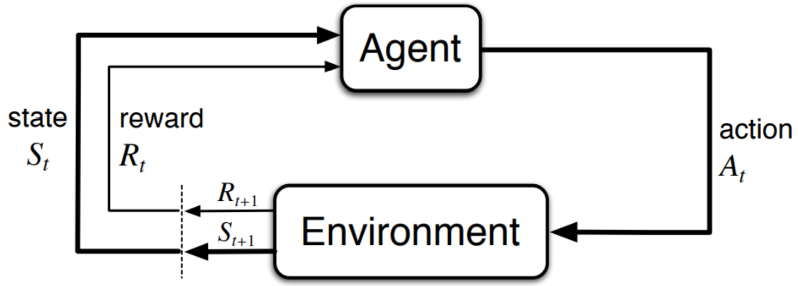


- action = "Anything the agent can do to achieve our result"
  - *example:*
    - moving chess pieces
    - moving balancing cart
    - and more





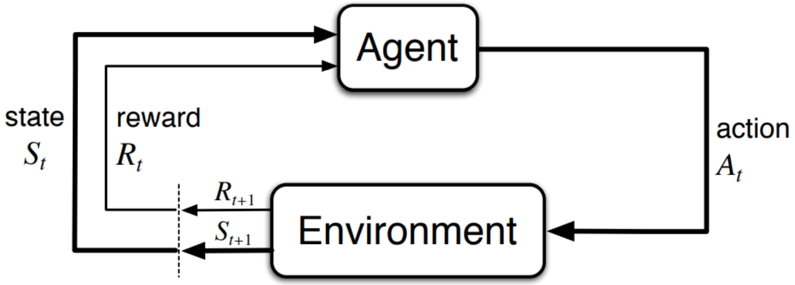
Policy $\pi(a|s)$



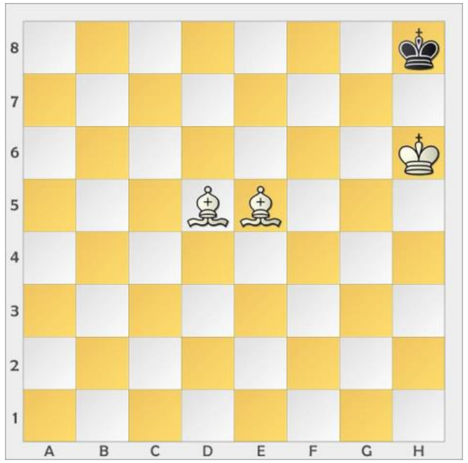
- policy = “probability of each possible action from a given state”
- probability to choose an action $A_t = a$ if in state $S_t = s$ at time t
  - ➢ probability distribution over $a \in A(s)$ for each $s \in S$

optimized due to agent's experience

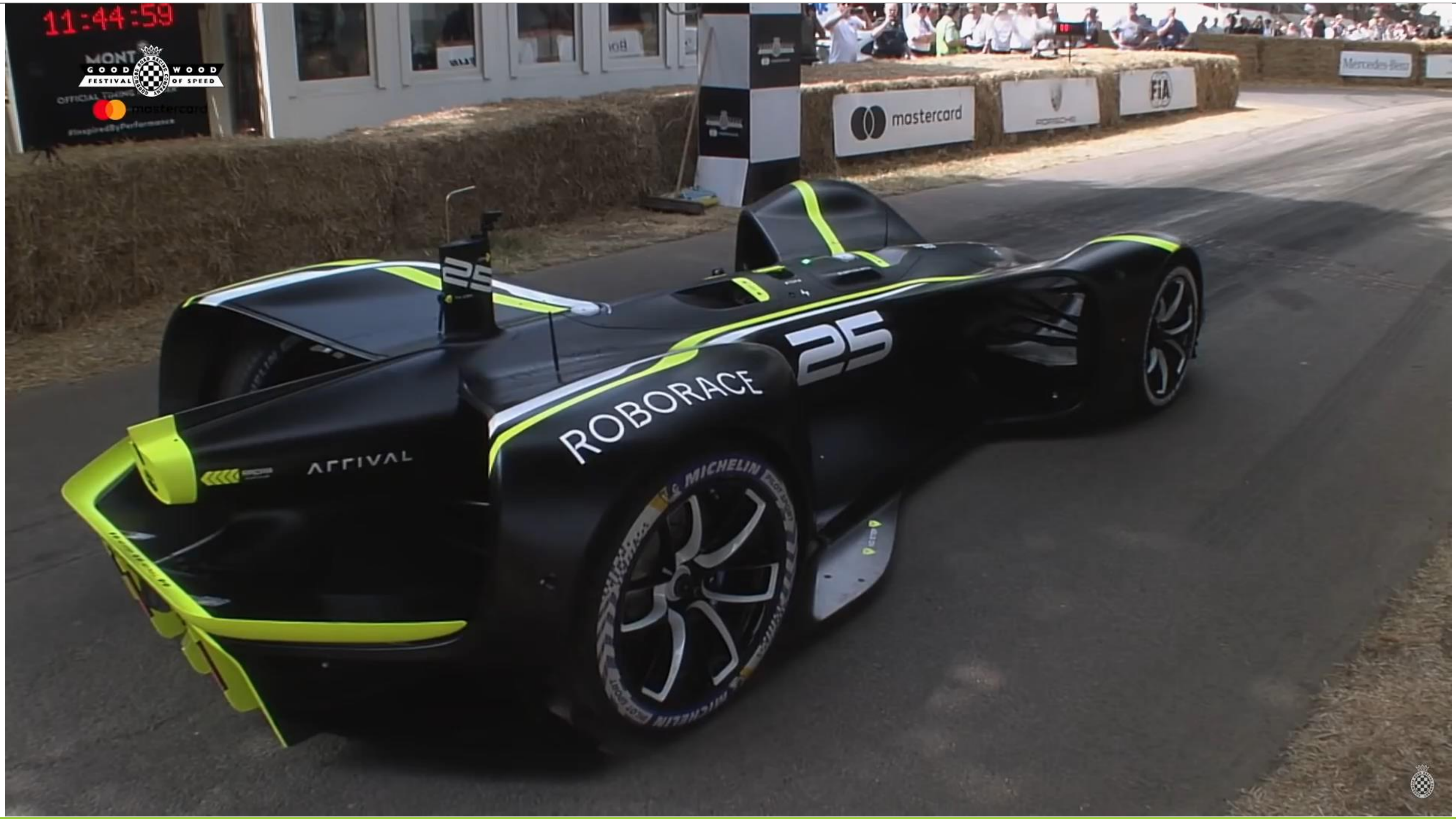
Reward space $\mathcal{R}$



- reward = "our way to tell the agent what we want"
- reward $R_t \in \mathbb{R}$
- agent wants to maximize the reward
  - **Important:**
    - provide rewards in such a way they achieve our goals
    - **don't** give rewards for subgoals

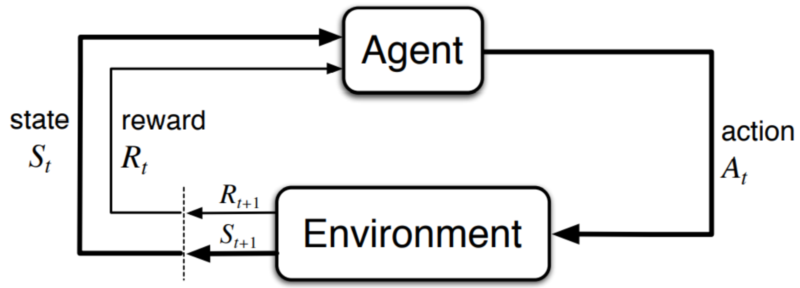


High reward


11:44:59
GOODWOOD
FESTIVAL OF SPEED
mastercard
FIA
Mercedes-Benz
PORSCHE
25
ROBORACE
ARRIVAL
MICHELIN

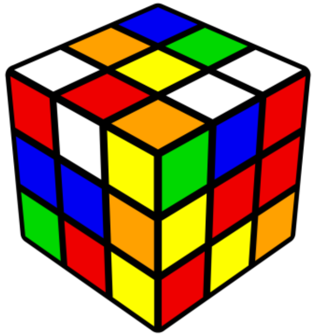
Low reward

Episodes



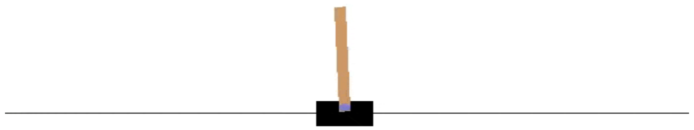
EPISODIC TASKS:

- every episode begins independently
- every episode ends in the same terminal state
- distinguish nonterminal state $\mathcal{S}$ from terminal state $\mathcal{S}^+$
- *example:*



CONTINUING TASKS:

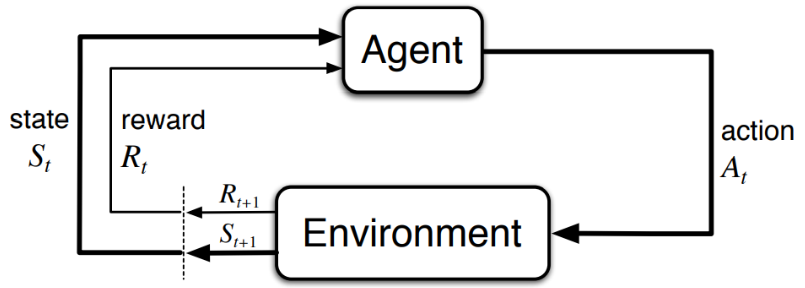
- $T = \infty$
- return could be infinite (see next slide)
  - use discounting return
- *example:*



Unified notation:

- $S_t \rightarrow S_{t,i}$
- State representation at time t of episode i

Return G_t



EPISODIC TASKS

- **Remember:** agent receives rewards after each “time” step
  - ➢ noted as $R_{t+1}, R_{t+2}, R_{t+3}, \ldots.$
- wish to maximize the expected return G_t
  - Simplest variation:
    - ➢ $G_t \equiv R_{t+1} + R_{t+2} + \cdots + R_T$ (T: final step)

CONTINUING TASKS

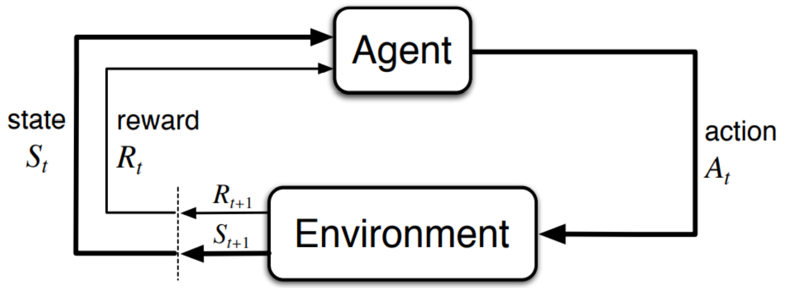
- sum discounted rewards:
  - ➢ $G_t \equiv R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
    $= R_{t+1} + \gamma G_{t+1}$
- $0 \le \gamma \le 1$: called discount rate
  - ➢ rewards received k times is worth only γ^{k-1} times

Unified notation:

- Add special *absorbing state*
  - ➢ Transition *only* on itself
    - ➢ Reward $R_t = 0$
- $G_t \equiv \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

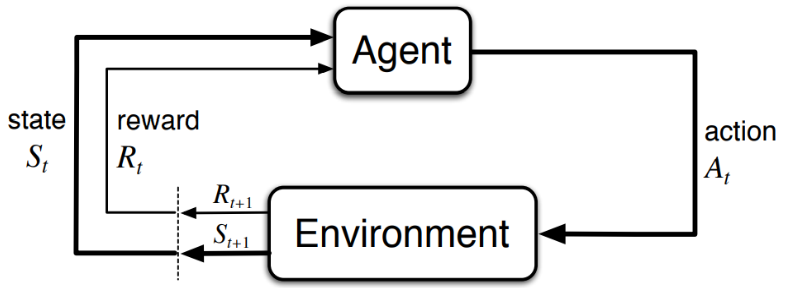
Optimization

Value functions



- tells us the possible future rewards
  - **Estimated from experience**
- defined with respect to policies
  - used to determine policy
- two kinds of value functions
  - state-value function $V_\pi(s)$
  - action-value function $q_\pi(s, a)$

State-value function $v_\pi(s)$

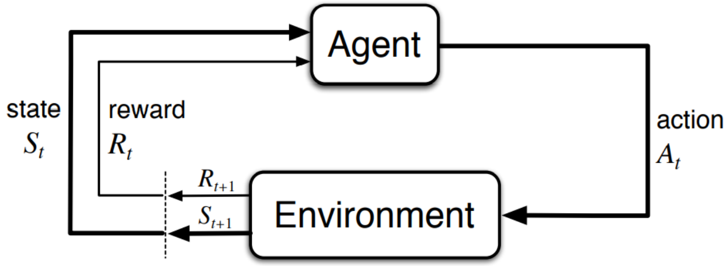


- value function of state s under policy π
- $v_\pi(s) \equiv \mathbb{E}_\pi(G_t|S_t = s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s]$
- $\mathbb{E}_\pi[\cdot]$ is the expected value of a random variable
  - By given policy π and time step t
- value of terminal state is always zero
- Problem:
  - Separate average for every state
  - ➢ Therefore:

$$v_\pi(s) \equiv \mathbb{E}_\pi[G_t|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma\, v_\pi(s')]$$

➢ So called *Bellman equation*

Action-value function $q_\pi(s, a)$

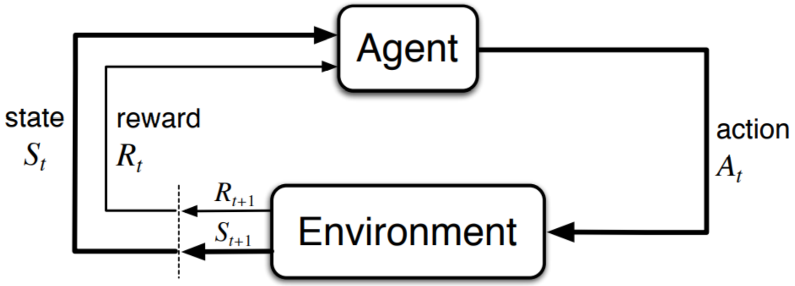


- value function of an action a in state s under policy π
- $q_\pi(s,a) \equiv \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$

➢ expected return starting in s, taking action a and afterwards policy π

➢ is the basis of the policy

Optimal solutions



- π is better than π', **if** expected return is greater or equal to π'
  - $\pi \geq \pi'$ **if** $v_{\pi}(s) \geq v_{\pi'}(s)$

- Optimal state-value function:

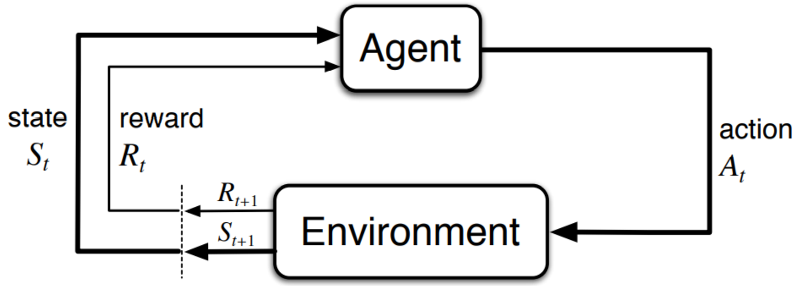
$$v_*(s) = \max_{\pi}\big(V_{\pi}(s)\big) \text{ for all } s \in S$$

- Optimal action-value function:

$$q_*(s,a) \equiv \max_{\pi} q_{\pi}(s,a)$$

- optimal policy is:

$$\begin{cases} \pi_*(a|s) = 1, & \text{if } \underset{a'}{\operatorname{argmax}}\, Q(s|a) \\ \pi_*(a|s) = 0, & else \end{cases}$$



Optimization

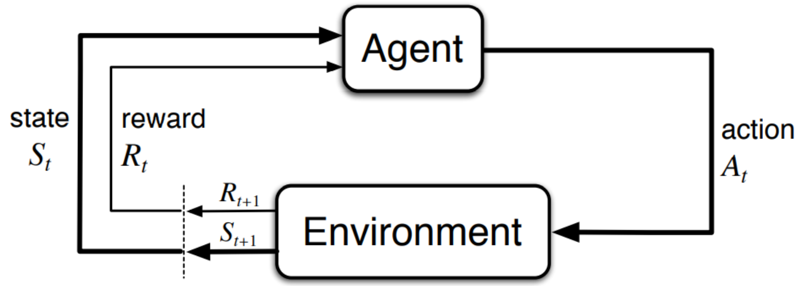
- solve Bellman equation
  - $v_\pi(s) \equiv \mathbb{E}_\pi[G_t|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma\, v_\pi(s')]$
  - Find optimal q-function

  - **Problem:**
    - Rarely directly useful
    - Solution relies on three assumptions
      1. Know the dynamics
      2. Enough computational recourses
      3. Markov property

rarely all true all the time

- critical aspects for alternatives
  - Computational power
  - Availability of memory

Optimization alternatives



Goal: Find "perfect" $Q_\pi(s, a)$

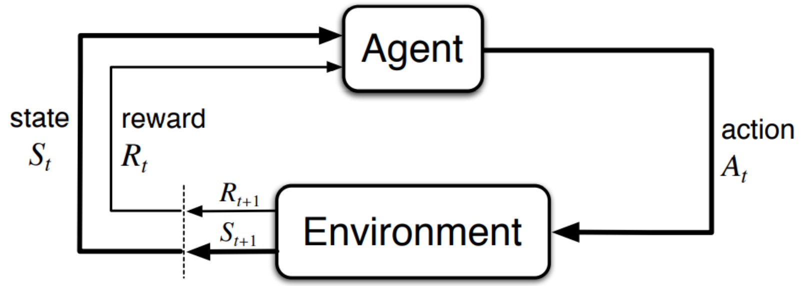
TASKS WITH SMALL, FINITE SETS

- approximations using arrays or tables
  - "tabular case"

TASKS WITH CONTINUOUS STATES

- use more compact parameterized function representation
  - approximating optimal behavior
    - Dismiss states with a low probability
  - Approximating optimal policies
    - More effort into learning good actions in frequently encountered states
    - Less effort into rarely encountered states

Watkins Q-learning



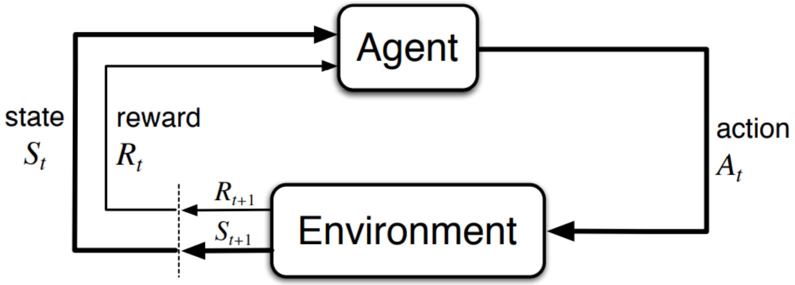
- *use:* action-value function Q(s,a)
- remember: Bellman equation
  - $v_\pi(s) \equiv \mathbb{E}_\pi[G_t | S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma\, v_\pi(s')]$
- *Define:* $Q^*(s,a) \equiv Q^{\pi^*}(s,a)$ for all s,a
- ***steps:***
  - Observe current state s_n
  - Select and perform a_n
  - Observe subsequent state s'
  - Receive immediate payoff r_n
  - update its Q_{n-1} values

Q-table		actions				
		a_1	…	a_n	…	a_m
states	s_1	0		0		0
	⋮					
	s_n			$\boxed{Q_n(s,a)}$		
	⋮					
	s_m	0		0		0

Q-updating rule:

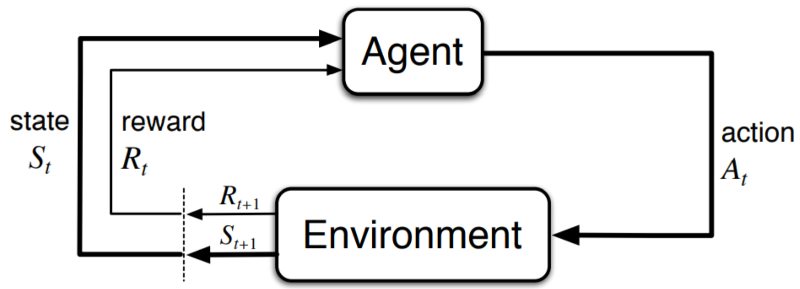
$$Q_n(s,a) = \begin{cases} (1-\alpha_n)Q_{n-1}(s,a) + \alpha_n[r_n + \gamma V_{n-1}(s'_n)] & \text{if } s=s_n,\ a=a_n \\ Q_{n-1}(s,a) & \text{otherwise} \end{cases}$$

Learning rate α



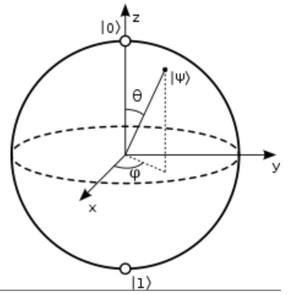
- Q-updating rule:
  - $Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \boldsymbol{\alpha}[r_i + \max_a[Q(s_{i+1}, a) - Q(s_i, a_i)]]$
- $\alpha \in (0,1)$
  - $\alpha \approx 1$: very fast learning
    - Necessary to slow down if Bellman error $\delta_t = r_i + \max_a Q(s_{i+1}, a) - Q(s_i, a_i)$
  - $\alpha \approx 0$: very slow learning

Exploration-exploitation dilemma



- necessary to avoid getting stuck in the in a local maximum of reward space
  - therefore, explore large parts of the RL state space
  - no exploration = agent repeats a given policy
    - Unclear if better policy exists
- **if** stuck in local maximum:
  - run multiple times with random starting conditions
    - Post select outcome
- RL solution nearly perfect
  - Fidelity close to true global optimal fidelity
- RL independent of initial conditions
  - **but** huge drop in fidelity if phases different

1. Exploratory training stage

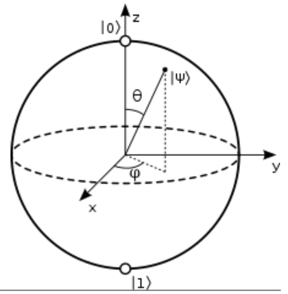


- exploits the current Q function to explore
- Amount of exploration set by “learning” temperature β_{RL}
  - $\beta_{RL} = 0$: random action
  - $\beta_{RL} = \infty$: greedy action
  - Respectively to current estimated of Q function
  - possible to determine policy:

$$\boxed{\pi(a|s) = \frac{e^{\beta_{RL} Q(s,a)}}{\sum_{a'} e^{\beta_{RL} Q(s,a')}}}$$

Number of episodes increases → Usage of β_{RL} decreases linearly

2. Replay training stage



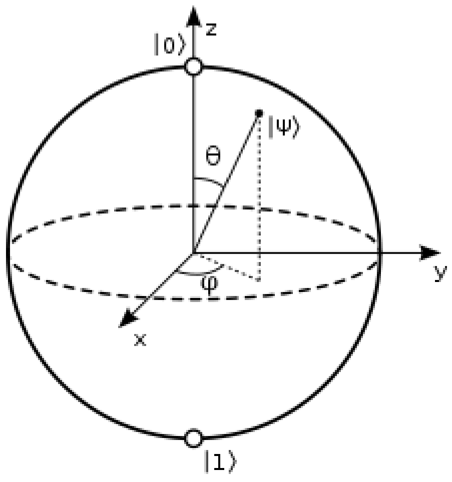
- replay best encountered protocol

- lasts 40 episodes

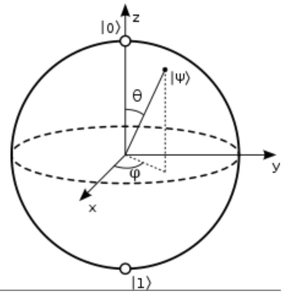
- take action according to softmax probability distribution based on values of Q-function
  - at each time step:
    - Look at Q(s,:) corresponding to all available actions
    - Compute $P(a) \sim \exp(\beta_{RL} Q(s,a))$

- agent will be biased toward the best encountered protocol
  - Improving until good fidelity

Qubit controlling



Quantum agent and environment



- Environment = $\{i\ \partial_t|\Psi(t)\rangle = H(t)|\Psi(t)\rangle, |\Psi(0)\rangle = |\Psi_i\rangle$

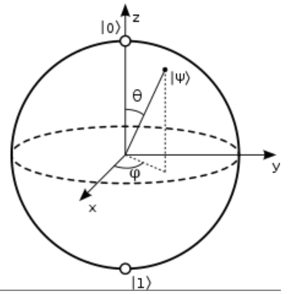
$$\boxed{H[h_x(t)] = -S^z - h_x(t)S^x}$$

  - is the Hamiltonian
    - Time dependence defined by the magnetic field $h_x(t)$
  - initial state $|\Psi_i\rangle$ at $h_x = -2$
  - target state $|\Psi_*\rangle$ at $h_x = 2$

  Ground state

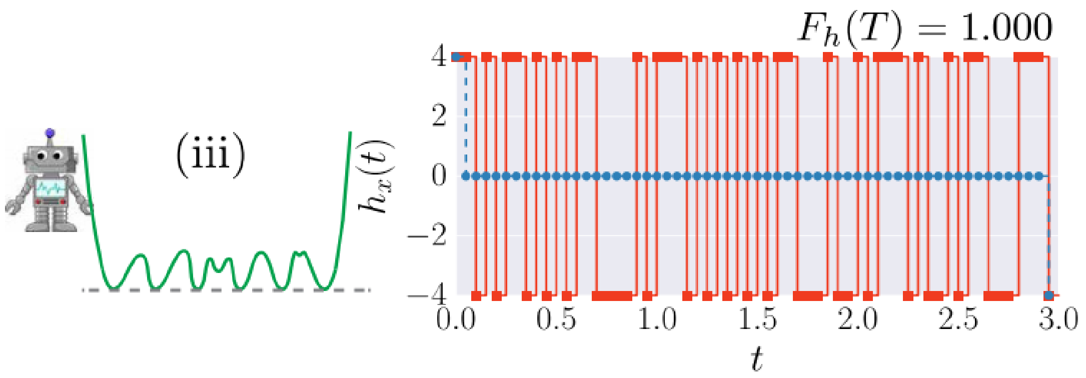
- agent constructs piecewise-constant protocols of duration T
  - agent chooses a drive protocol strength $h_x(t)$ at each time $t = j\ \delta t,\ with\ j = \left\{0,1,\cdots,\frac{T}{\delta t}\right\}$

Restrictions

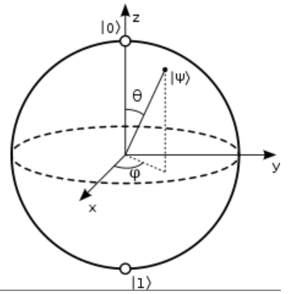


- no access to infinite control fields
  - restrict to field $h_x(t) \in \{-4,4\}$
  - use equal spaced tilings along the entire range of $h_x(t) \in [-4,4]$

- restrict the RL algorithm to the family of bang-bang protocols
  - protocols that switch abruptly between two states

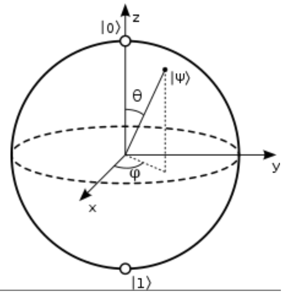


State space $\mathcal{S}$



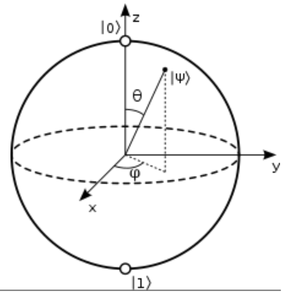
- $S = \{s = [t, h_x(t)]\}$
  - All tuples $[t, h_x(t)]$ of time t and corresponding magnetic field $h_x(t)$
- **model free !**
  - **Discrete states**
  - agent avoids difficulties produced by theoretical notions
  - 'time' t shows us where episodes ends
- even though only one control field available protocols grows exponentially with δt^{-1}

Action space $\mathcal{A}$



- consists of all jumps δh_x in the protocol $h_x(t)$
- protocols constructed as piece-wise constant functions
- restrict the available actions in every state s
  - $h_x(t) \in \{-4,4\}$

Reward space $\mathcal{R}$

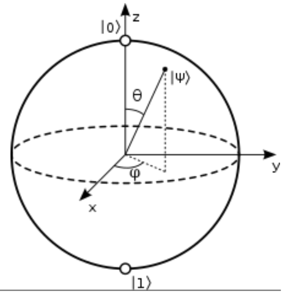


- real number in the interval [0,1]
- rewards given at the end of each episode to:
    - $r(t) = \begin{cases} 0 \\ F_h(T) = |\langle \Psi_* | \Psi(T) \rangle|^2 \end{cases}$

- we are not interested in the system during its evolution
- all that matters is to maximize the final fidelity $F_h(T)$

- For fixed protocol duration T use the infidelity $I_h(T) = 1 - F_h(T)$
    - Global minimum corresponds to optimal driving protocol

Protocol construction algorithm



- *algorithm:*
- start in the initial RL state
- Take action
- next RL state
- ➢ initial q.s evolved forward in time

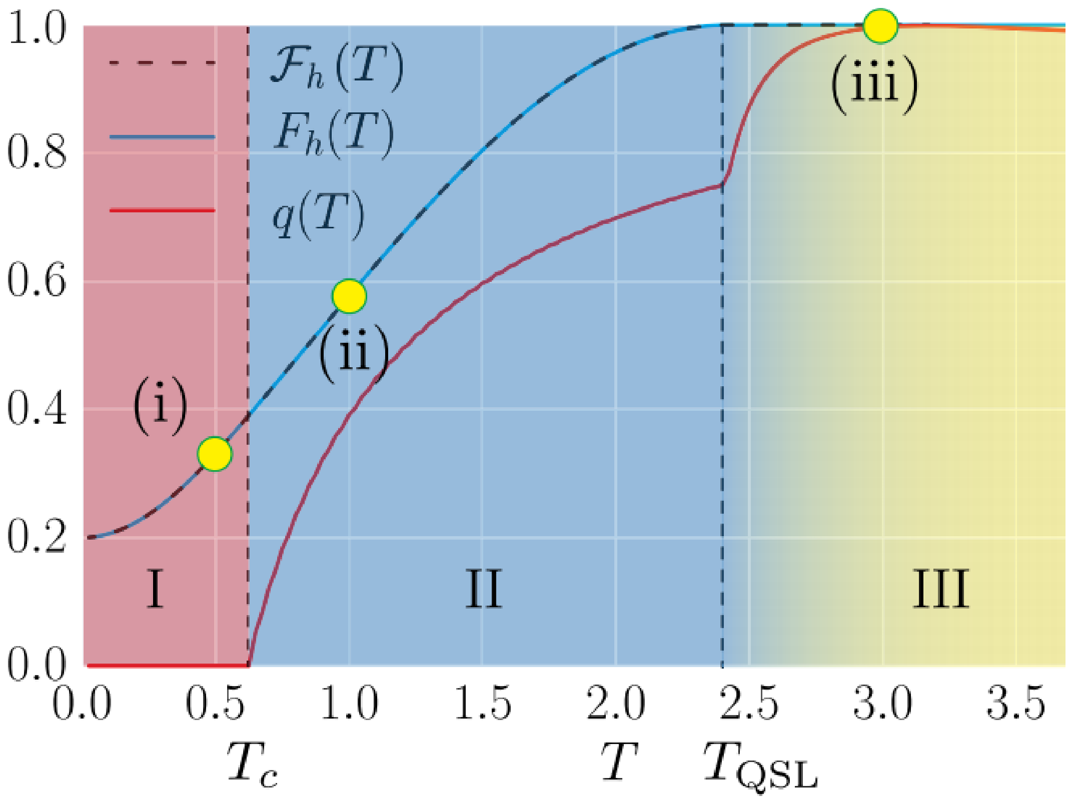
- *example:*
- $s_0 = (t = 0, h_x = -4)$
- $a = \delta h_x = 8$
- $s_1 = (\delta t, +4)$
- $t_0 = 0 \ \rightarrow t_1 = \delta t$
  - $|\Psi(\delta t)\rangle = e^{-i*H[h_x=4]\delta t}|\Psi_i\rangle$

➢ Compute reward and update Q function

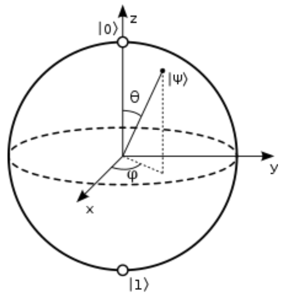
➢ Repeat until t = T

$$\boxed{s_0 \rightarrow a_0 \rightarrow r_0 \rightarrow s_1 \rightarrow a_1 \rightarrow r_1 \rightarrow s_1 \rightarrow \cdots \rightarrow s_{N_T}}$$

The 3 phases of reinforcement learning



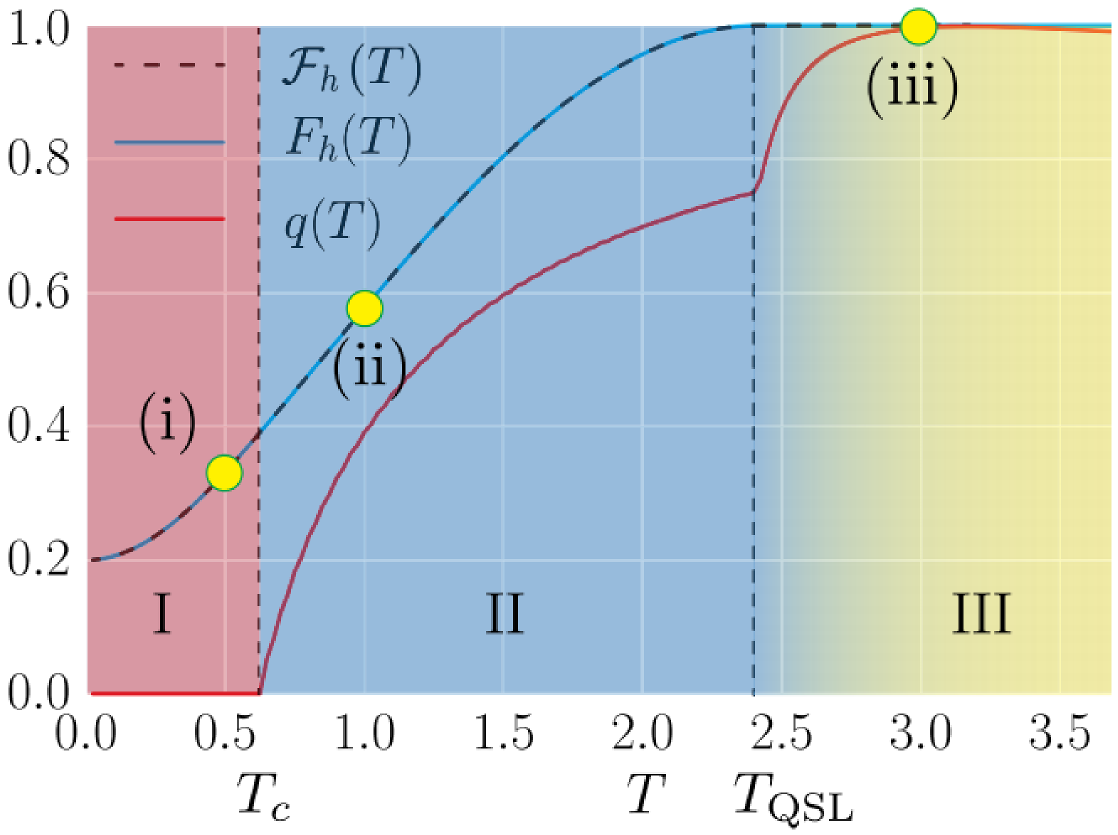
The correlator

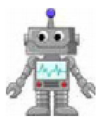


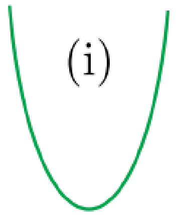
- total protocol duration T = fixed
- the infidelity: $h_x(t) \longmapsto I_h(T) = 1 - F_h(T)$
  - ➢ global minimum is the optimal driving protocol

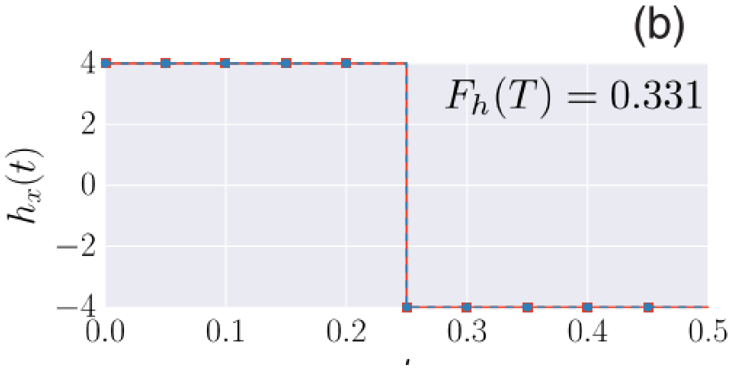
$$\boxed{q(T) = \frac{1}{16\, N_T} \sum_{j=1}^{N_T} \overline{\left\{h_x(j\delta t) - \overline{h_x(j\delta t)}\right\}^2}}$$

- where $\overline{h_x}(t) = \frac{1}{N_{real}} \sum_{\alpha=1}^{N_{real}} h_x^{\alpha}(t)$
- q(T) = correlator between the infidelity minima
  - **If** $\{h_x^a(t)\}_{a=1}^{N_{real}}$ are all uncorrelated
    - ➢ $\overline{h_x(t)} \equiv 0$, thus q(T) = 1
  - **Only** one minimum
    - ➢ $\overline{h_x}(t) \equiv h_x(t)$ and $q(T) = 0$







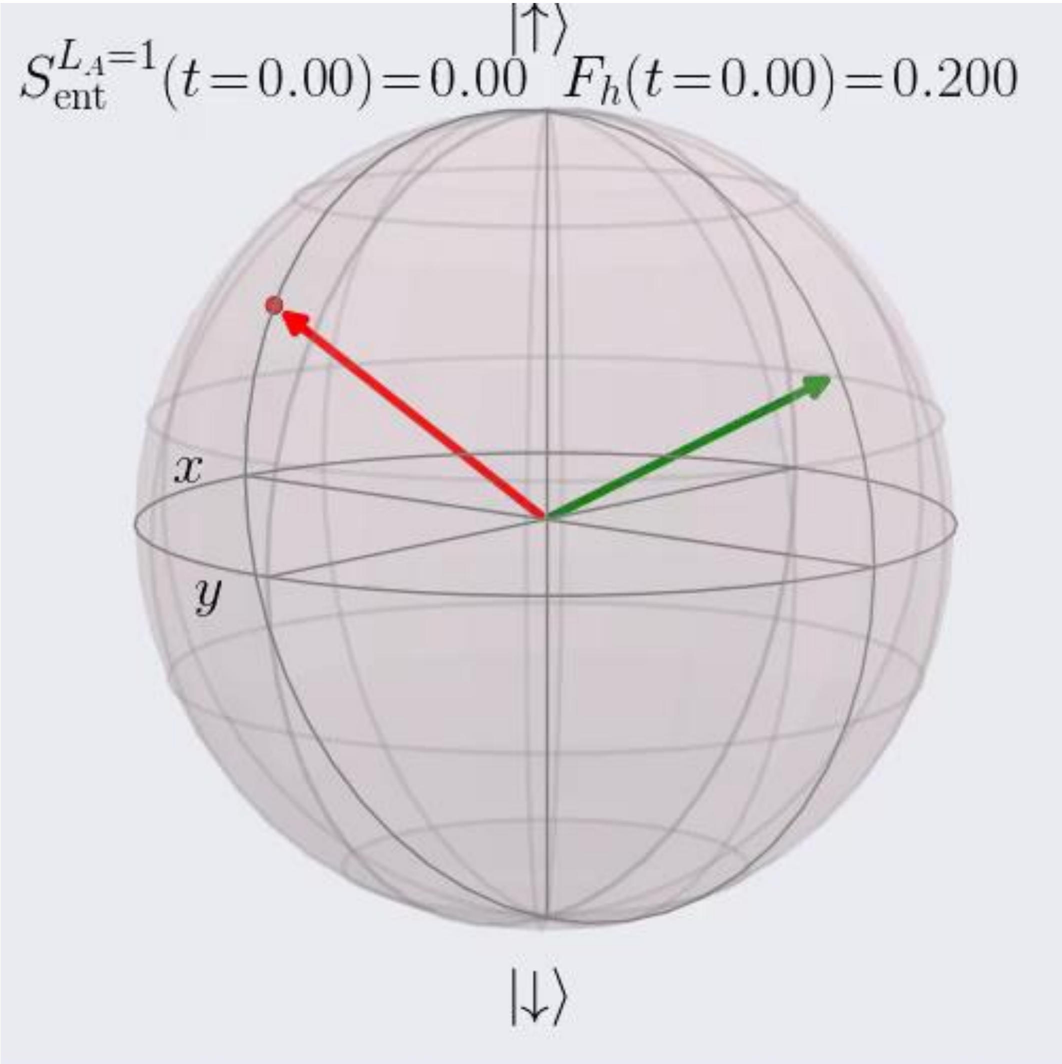


The control problem I

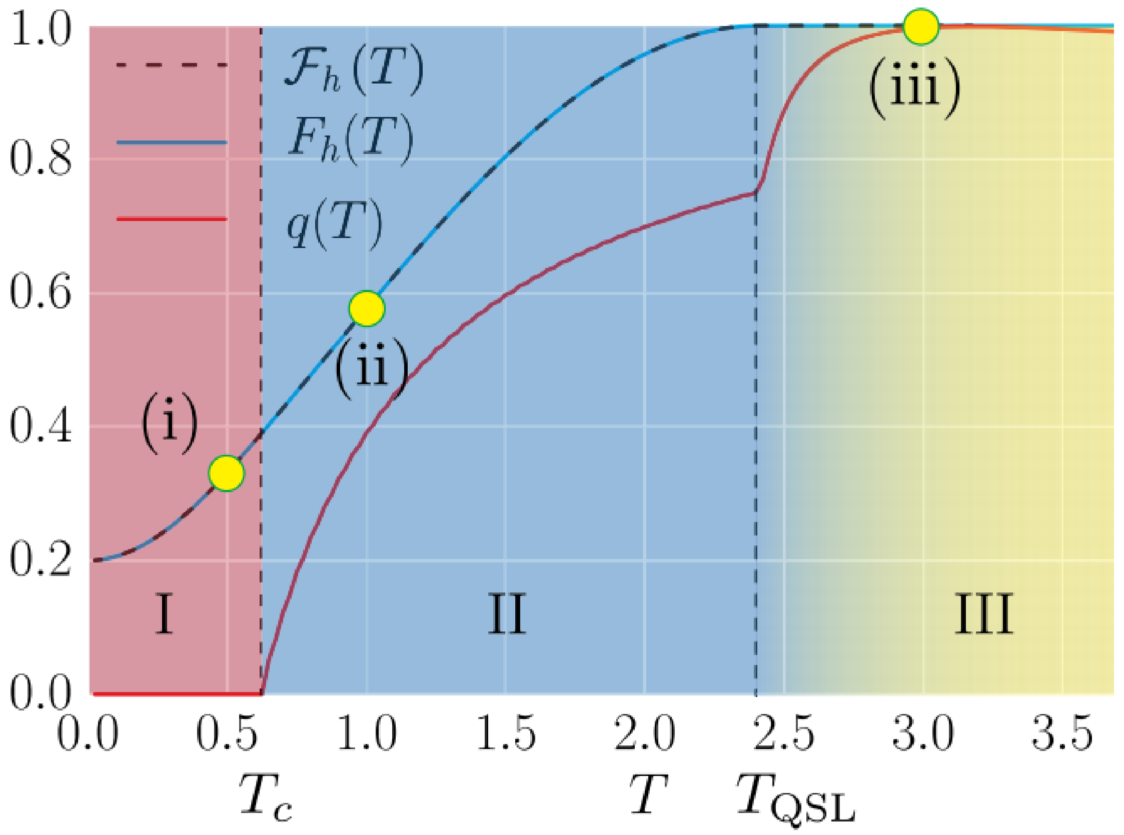
1st phase @ $T < T_c \approx 0.6$:

- *called: over constrained phase*
- *unique optimal protocol*
  - $q(T) \equiv 0 \longrightarrow$ infidelity landscape convex
  - fidelity can be limited
- $T_c \to 0$ for $|h_x| \to \infty$

➢ Precession speed towards equator dependent on maximum possible allowed field strength



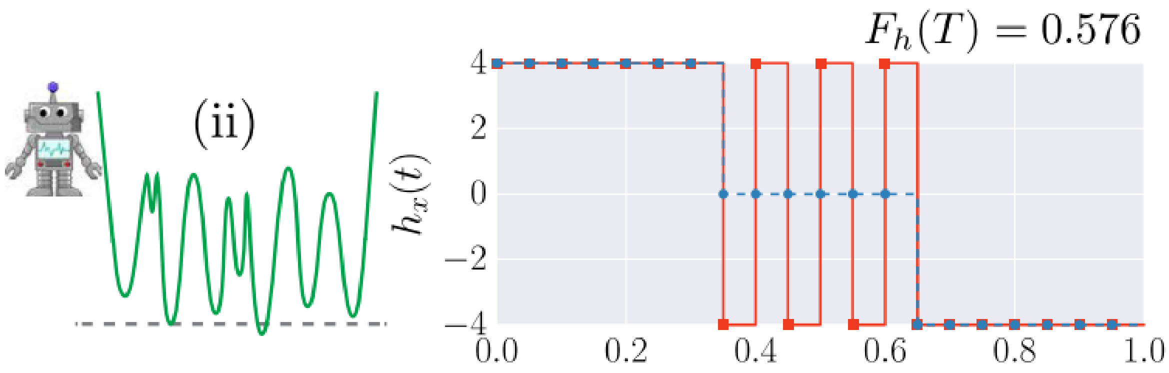
The overcon-strained phase:

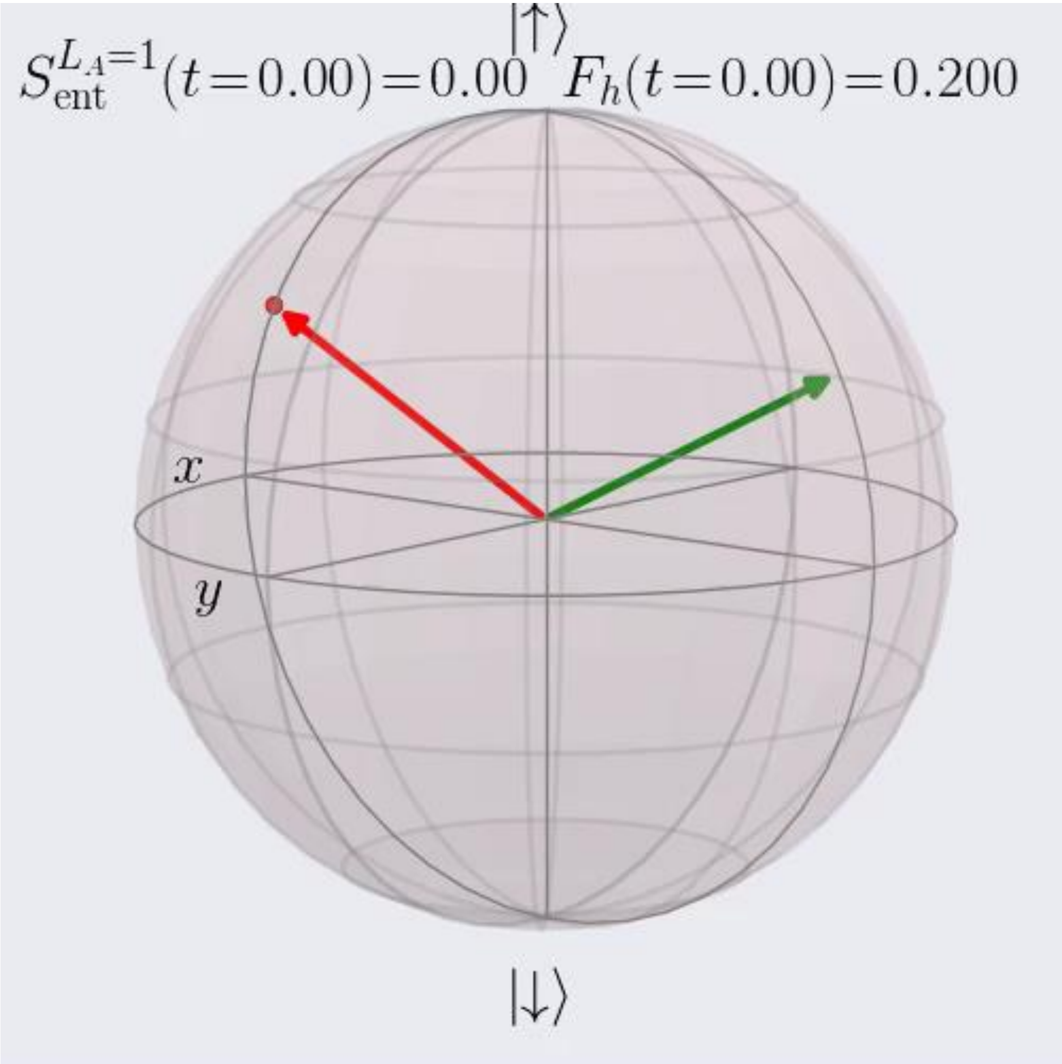


The control problem II

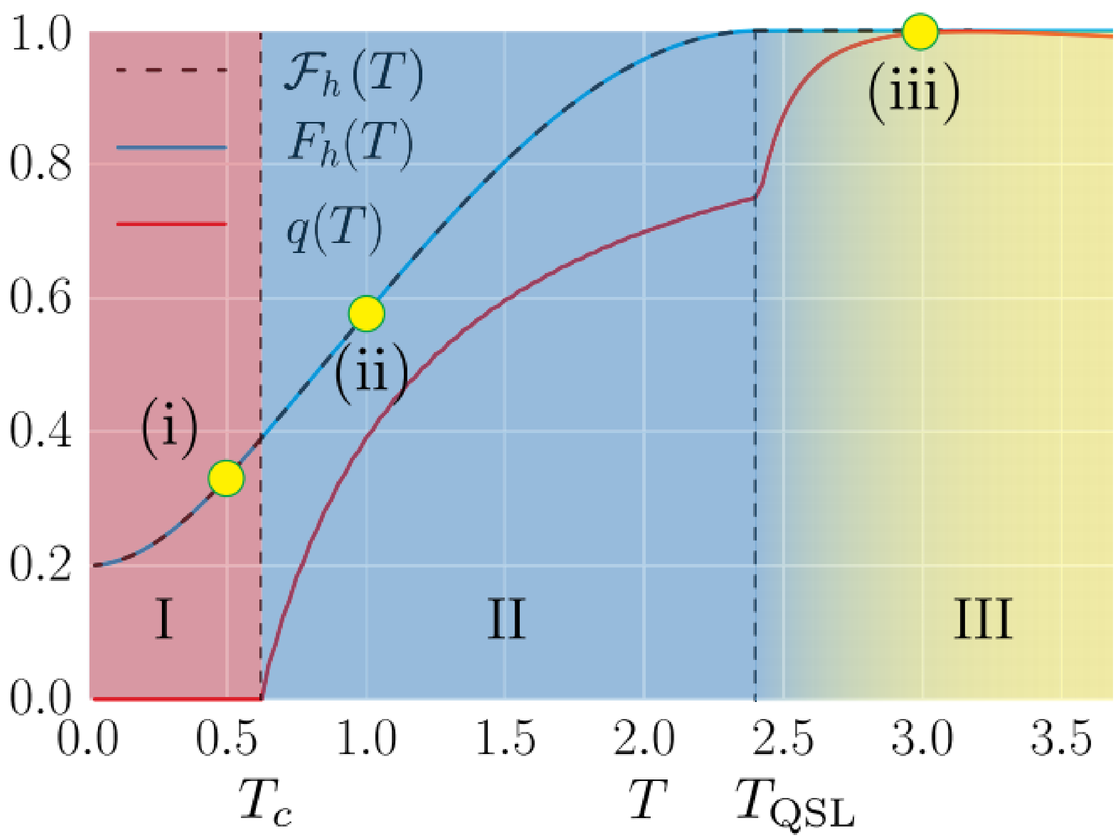
2nd phase @ $T_c < T < T_{QSL}$:

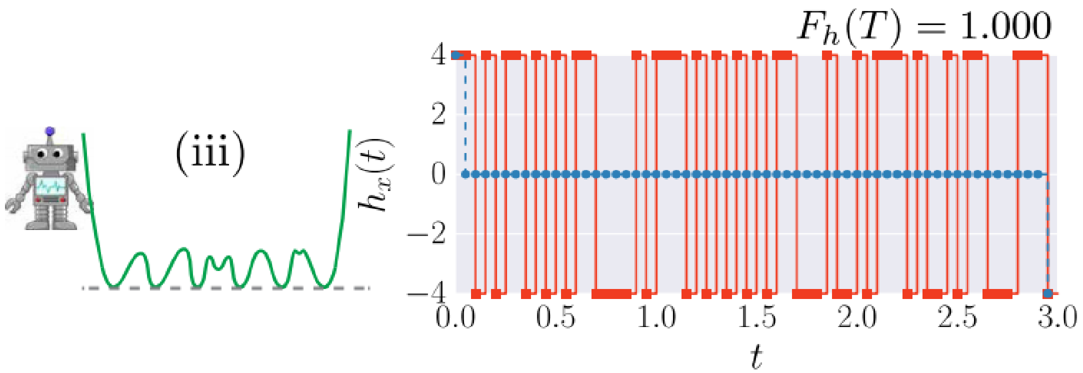
- *called:* glassy phase
- infidelity landscape won't form a minima corresponding to protocols of unit fidelity





The glassy phase:

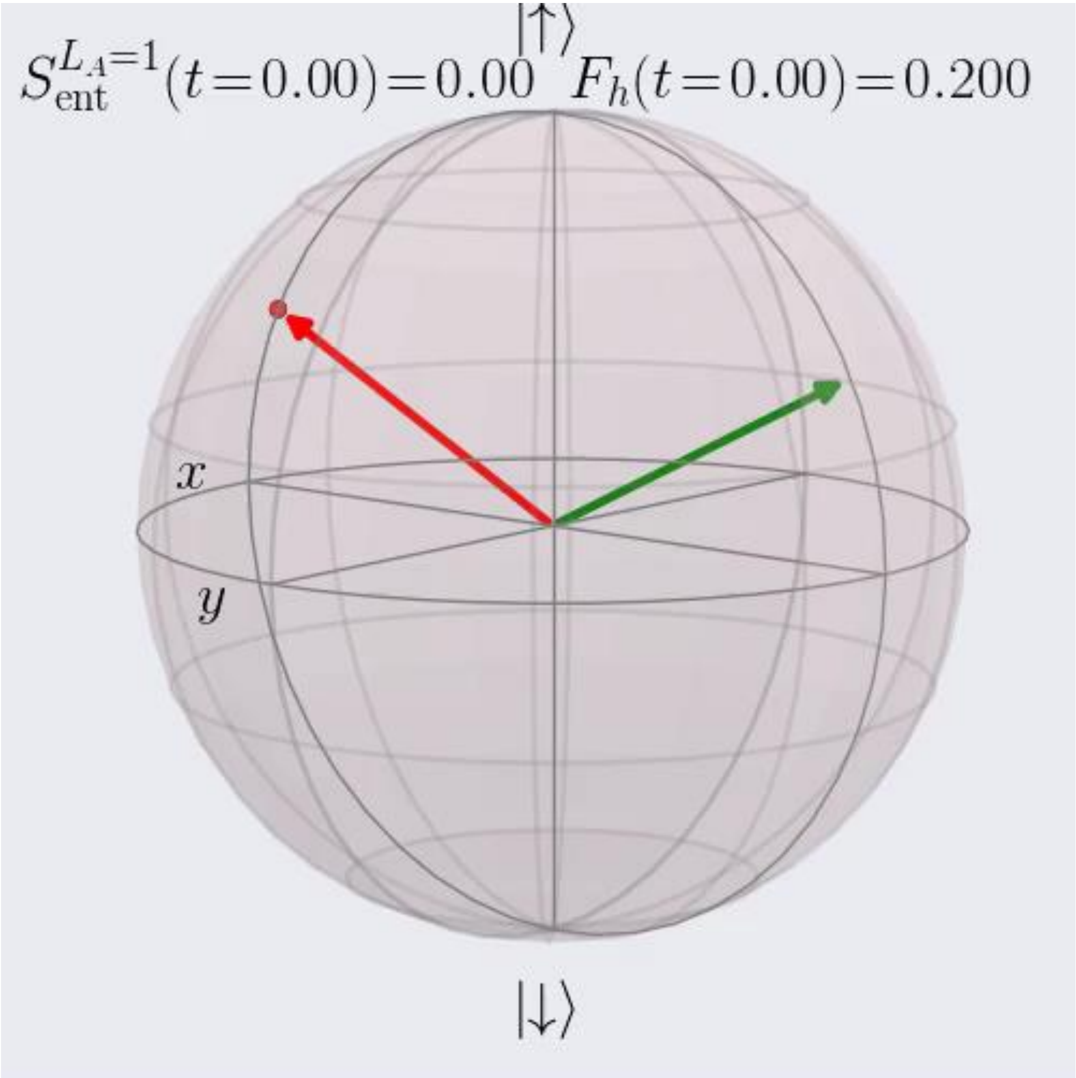




The control problem III

3rd phase @ $T > T_{QSL} \approx 2.4$:

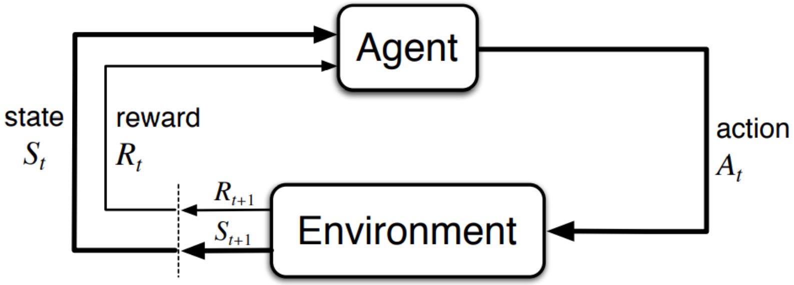
- *called: controllable phase*
- infinitely many protocols constructable
  - all prepare target state with unit fidelity



The control-able phase:

Summary

FINITE MARKOV DECISION PROCESS

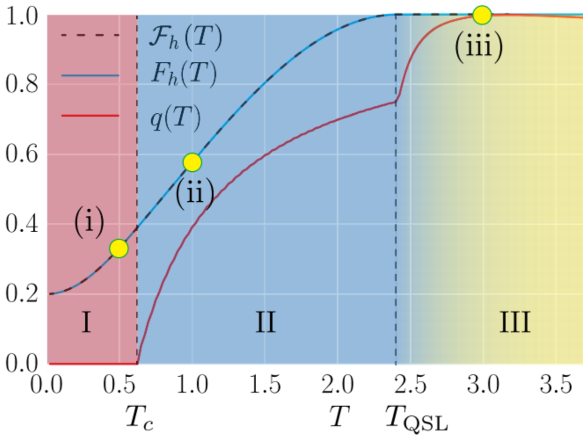


Policy: probability of each possible action from a given state

Value function: estimates future rewards depending on experience and policy

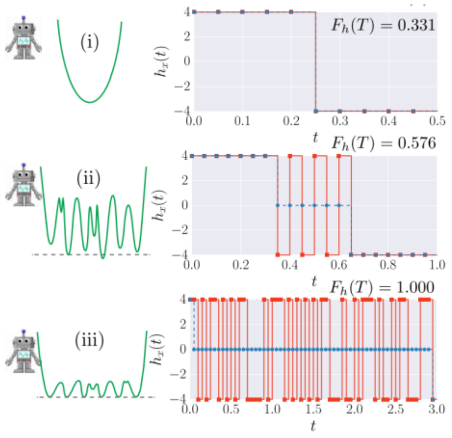
Optimization: Watkins Q-learning

QUBIT CONTROLLING



3 phases:

- Over constrained phase
- Glassy phase
- Controllable phase



Bibliography

- Reinforcement Learning in Different Phases of Quantum Control; Marin Bukov, Alexandre G. R. Day,† Dries Sels, Phillip Weinberg, Anatoli Polkovnikov and Pankaj Mehta
- Reinforcement Learning, second edition: An Introduction (Adaptive Computation and Machine Learning series) (2. Aufl.); Sutton, R. S. & Barto, A. G. (2018). Bradford Books.
- Technical Note: Q-Learning; Christopher J.C.H. Watkins & Peter Dayan, 1992 Kluwer Academic Publishers, Boston.

Pictures

https://www.freeimages.com/de/photo/rubix-cube-solved-1196475

https://www.chess-international.com/?p=754,

https://de.wikipedia.org/wiki/Schach

https://www.spielezar.ch/blog/spielregeln

https://medium.com/@tuzzer/cart-pole-balancing-with-q-learning-b54c6068d947

https://de.freepik.com/fotos-vektoren-kostenlos/pack-robot

https://www.youtube.com/watch?v=x4fdUx6d4QM

https://www.youtube.com/watch?v=QtVbch-02Fs