A Tutorial on Optimal Control and Reinforcement Learning methods for Quantum Technologies

arXiv:2112.07453

Luigi Giannelli[1,2], Pierpaolo Sgroi[3], Jonathon Brown[3], Gheorghe Sorin Paraoanu[4], Mauro Paternostro[3], Elisabetta Paladino[1,2,5], and Giuseppe Falci[1,2,5]

[1]Dipartimento di Fisica e Astronomia “Ettore Majorana”, Universitá di Catania,
[2]CNR-IMM, Catania (University) Unit,
[3]CTAMOP, School of Mathematics and Physics, Queens University,
[4]QTF Centre of Excellence, Department of Applied Physics, Aalto University School of Science,
[5]INFN, Sez. Catania.

Pgi-controlclub, 2022-03-11

motivation

Quantum control is central to most quantum technologies (computing, simulation, metrology, ...)

Quantum Optimal Control is a widely used tool for the development of quantum technologies

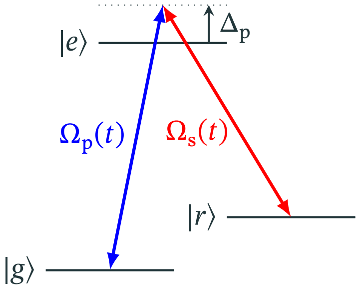
Reinforcement Learning has huge success in robotics and games, and offers a direct approach to control problems

outline

1. population transfer in three-level systems and STIRAP
2. “super” *very* <u>short</u> mention of **Optimal Control** and application to 3LS
3. “brief” idea of **Reinforcement Learning** and application to 3LS
4. discussion and conclusions

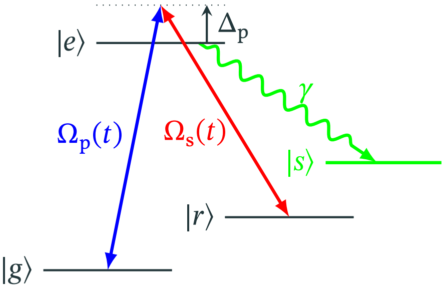
population transfer in three-level systems and STIRAP

population transfer in a three-level system



$$\frac{H(t)}{\hbar} = \Delta_\mathrm{p}|e\rangle\langle e| + \frac{\Omega_\mathrm{p}(t)}{2}(|g\rangle\langle e| + |e\rangle\langle g|) + \\ + \frac{\Omega_\mathrm{s}(t)}{2}(|e\rangle\langle r| + |r\rangle\langle e|)$$

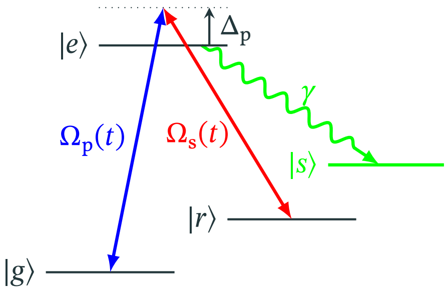
population transfer in a three-level system



$$\frac{H(t)}{\hbar} = \Delta_\mathrm{p}|e\rangle\langle e| + \frac{\Omega_\mathrm{p}(t)}{2}(|g\rangle\langle e| + |e\rangle\langle g|) + \\ + \frac{\Omega_\mathrm{s}(t)}{2}(|e\rangle\langle r| + |r\rangle\langle e|)$$

$$\dot{\rho}(t) = -\frac{\mathrm{i}}{\hbar}[H(t), \rho(t)] + \frac{\gamma}{2}(2|s\rangle\langle e|\rho(t)|e\rangle\langle s| - |e\rangle\langle e|\rho(t) - \rho(t)|e\rangle\langle e|)$$

population transfer in a three-level system

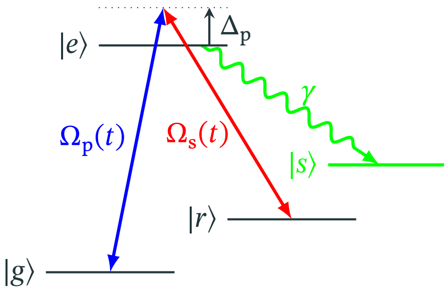


$$\frac{H(t)}{\hbar} = \Delta_p|e\rangle\langle e| + \frac{\Omega_p(t)}{2}(|g\rangle\langle e| + |e\rangle\langle g|) + \\ + \frac{\Omega_s(t)}{2}(|e\rangle\langle r| + |r\rangle\langle e|)$$

$$\dot{\rho}(t) = -\frac{i}{\hbar}[H(t), \rho(t)] + \frac{\gamma}{2}(2|s\rangle\langle e|\rho(t)|e\rangle\langle s| - |e\rangle\langle e|\rho(t) - \rho(t)|e\rangle\langle e|)$$

$$\boxed{\rho(0) = |g\rangle\langle g| \quad \longrightarrow \quad \rho(T) = |r\rangle\langle r|}$$

population transfer in a three-level system



$$\frac{H(t)}{\hbar} = \Delta_{\mathrm{p}}|e\rangle\langle e| + \frac{\Omega_{\mathrm{p}}(t)}{2}(|g\rangle\langle e| + |e\rangle\langle g|) + \frac{\Omega_{\mathrm{s}}(t)}{2}(|e\rangle\langle r| + |r\rangle\langle e|)$$

$$\dot{\rho}(t) = -\frac{\mathrm{i}}{\hbar}[H(t), \rho(t)] + \frac{\gamma}{2}(2|s\rangle\langle e|\rho(t)|e\rangle\langle s| - |e\rangle\langle e|\rho(t) - \rho(t)|e\rangle\langle e|)$$

$$\boxed{\rho(0) = |g\rangle\langle g| \longrightarrow \rho(T) = |r\rangle\langle r|}$$

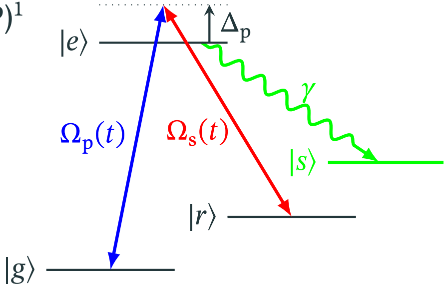
fidelity

$$\mathcal{F} = \mathrm{Tr}\{\rho(T)|r\rangle\langle r|\}$$

population transfer in a three-level system - STIRAP

STimulated Raman Adiabatic Passage (STIRAP)[1]

- adiabatic protocol
- population of the lossy state $|e\rangle$ low
- $\mathcal{F} \approx 1$



[1] J. R. Kuklinski, U. Gaubatz, F. T. Hioe, K. Bergmann, Phys. Rev. A 40 (11) (1989),
K. Bergmann, H. Theuer, B. Shore, Reviews of Modern Physics 70 (3) (1998),
N. V. Vitanov, A. A. Rangelov, B. W. Shore, K. Bergmann, Reviews of Modern Physics 89 (1) (2017).

Adiabatic Theorem

Given a time dependent Hamiltonian $H_0(t)$ and its instantaneous eigenstates $|n(t)\rangle$

$$H_0(t)|n(t)\rangle = E_n(t)|n(t)\rangle,$$

[2] $\hbar|\langle n(t)|\partial_t m(t)\rangle| \ll |E_n(t) - E_m(t)|, \forall m \neq n.$

Adiabatic Theorem

Given a time dependent Hamiltonian $H_0(t)$ and its instantaneous eigenstates $|n(t)\rangle$

$$H_0(t)|n(t)\rangle = E_n(t)|n(t)\rangle,$$

the solution of the Schrodinger equation $i\hbar\frac{\partial|\psi(t)\rangle}{\partial t} = H_0(t)|\psi(t)\rangle$

in general is $|\psi(t)\rangle = \sum_n c_n(t)|n(t)\rangle, \quad \sum_n |c_n(t)|^2 = 1.$

[2] $\hbar|\langle n(t)|\partial_t m(t)\rangle| \ll |E_n(t) - E_m(t)|, \forall m \neq n.$

Adiabatic Theorem

Given a time dependent Hamiltonian $H_0(t)$ and its instantaneous eigenstates $|n(t)\rangle$

$$H_0(t)|n(t)\rangle = E_n(t)|n(t)\rangle,$$

the solution of the Schrodinger equation $i\hbar\frac{\partial|\psi(t)\rangle}{\partial t} = H_0(t)|\psi(t)\rangle$

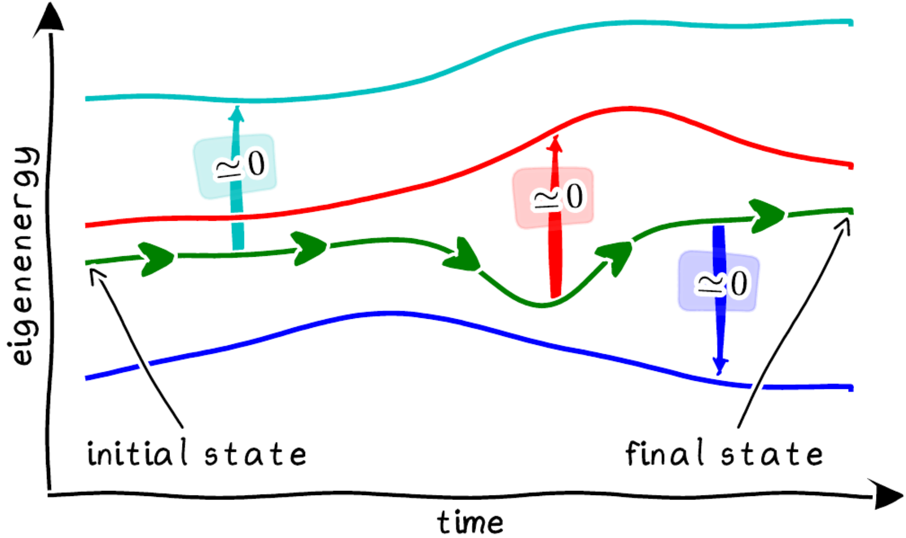
in general is $|\psi(t)\rangle = \sum_n c_n(t)|n(t)\rangle, \quad \sum_n |c_n(t)|^2 = 1.$

If $H_0(t)$ is slowly varying[2] and the initial state is an eigenstate, ie $|\psi(t_\mathrm{i})\rangle = |m(t_\mathrm{i})\rangle$, then

$$|\psi(t)\rangle \simeq \mathrm{e}^{\mathrm{i}\alpha_m(t)}|m(t)\rangle, \quad \forall t$$
$$\text{i.e. } c_n(t) \simeq \mathrm{e}^{\mathrm{i}\alpha_m(t)}\delta_{mn}.$$

[2] $\hbar|\langle n(t)|\partial_t m(t)\rangle| \ll |E_n(t) - E_m(t)|, \forall m \neq n.$

adiabatic following of an instantaneous eigenstate



population transfer in a three-level system - STIRAP

the three-level Hamiltonian we consider is $\frac{H(t)}{\hbar} = \frac{1}{2}\begin{pmatrix} 0 & \Omega_{\mathrm{p}}(t) & 0 \\ \Omega_{\mathrm{p}}(t) & 2\Delta_{\mathrm{p}} & \Omega_{\mathrm{s}}(t) \\ 0 & \Omega_{\mathrm{s}}(t) & 0 \end{pmatrix}$

population transfer in a three-level system - STIRAP

the three-level Hamiltonian we consider is $\frac{H(t)}{\hbar} = \frac{1}{2}\begin{pmatrix} 0 & \Omega_{\mathrm{p}}(t) & 0 \\ \Omega_{\mathrm{p}}(t) & 2\Delta_{\mathrm{p}} & \Omega_{\mathrm{s}}(t) \\ 0 & \Omega_{\mathrm{s}}(t) & 0 \end{pmatrix}$

and its instantaneous eigenstates are

$$|a_0(t)\rangle = \begin{pmatrix} \cos\theta \\ 0 \\ -\sin\theta \end{pmatrix} \quad |a_-(t)\rangle = \begin{pmatrix} \sin\theta\cos\phi \\ -\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \quad |a_+(t)\rangle = \begin{pmatrix} \sin\theta\sin\phi \\ \cos\phi \\ \cos\theta\sin\phi \end{pmatrix}$$

with

$$\tan\theta(t) = \frac{\Omega_p(t)}{\Omega_s(t)} \qquad \tan\phi(t) = \frac{\sqrt{\Omega_p(t)^2 + \Omega_s(t)^2}}{\Delta_p + \sqrt{\Delta_p^2 + \Omega_p(t)^2 + \Omega_s(t)^2}}$$

population transfer in a three-level system - STIRAP

the three-level Hamiltonian we consider is $\frac{H(t)}{\hbar} = \frac{1}{2}\begin{pmatrix} 0 & \Omega_{\mathrm{p}}(t) & 0 \\ \Omega_{\mathrm{p}}(t) & 2\Delta_{\mathrm{p}} & \Omega_{\mathrm{s}}(t) \\ 0 & \Omega_{\mathrm{s}}(t) & 0 \end{pmatrix}$

and its instantaneous eigenstates are

$$\boxed{|a_0(t)\rangle = \begin{pmatrix} \cos\theta \\ 0 \\ -\sin\theta \end{pmatrix}} \quad |a_-(t)\rangle = \begin{pmatrix} \sin\theta\cos\phi \\ -\sin\phi \\ \cos\theta\cos\phi \end{pmatrix} \quad |a_+(t)\rangle = \begin{pmatrix} \sin\theta\sin\phi \\ \cos\phi \\ \cos\theta\sin\phi \end{pmatrix}$$

with

$$\boxed{\tan\theta(t) = \frac{\Omega_p(t)}{\Omega_s(t)}} \qquad \tan\phi(t) = \frac{\sqrt{\Omega_p(t)^2 + \Omega_s(t)^2}}{\Delta_p + \sqrt{\Delta_p^2 + \Omega_p(t)^2 + \Omega_s(t)^2}}$$

population transfer in a three-level system - STIRAP

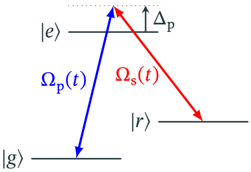
consider the instantaneous eigenstate $|a_0(t)\rangle$ (the *dark state*)

$$|a_0(t)\rangle = \begin{pmatrix} \cos\theta(t) \\ 0 \\ -\sin\theta(t) \end{pmatrix} = \cos\theta(t)|g\rangle - \sin\theta(t)|r\rangle, \qquad \tan\theta(t) = \frac{\Omega_p(t)}{\Omega_s(t)}$$

population transfer in a three-level system - STIRAP

consider the instantaneous eigenstate $|a_0(t)\rangle$ (the *dark state*)

$$|a_0(t)\rangle = \begin{pmatrix} \cos\theta(t) \\ 0 \\ -\sin\theta(t) \end{pmatrix} = \cos\theta(t)|g\rangle - \sin\theta(t)|r\rangle, \qquad \tan\theta(t) = \frac{\Omega_p(t)}{\Omega_s(t)}$$



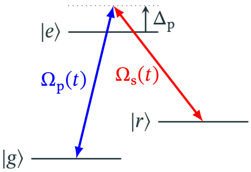
if the pulses are counterintuitively ordered

$$\lim_{t\to t_\mathrm{i}} \frac{\Omega_p(t)}{\Omega_s(t)} = 0, \quad \lim_{t\to t_\mathrm{f}} \frac{\Omega_s(t)}{\Omega_p(t)} = 0 \quad \Longrightarrow \quad \lim_{t\to t_\mathrm{i}} \theta(t) = 0, \quad \lim_{t\to t_\mathrm{f}} \theta(t) = \frac{\pi}{2}$$

population transfer in a three-level system - STIRAP

consider the instantaneous eigenstate $|a_0(t)\rangle$ (the *dark state*)

$$|a_0(t)\rangle = \begin{pmatrix} \cos\theta(t) \\ 0 \\ -\sin\theta(t) \end{pmatrix} = \cos\theta(t)|g\rangle - \sin\theta(t)|r\rangle, \qquad \tan\theta(t) = \frac{\Omega_p(t)}{\Omega_s(t)}$$



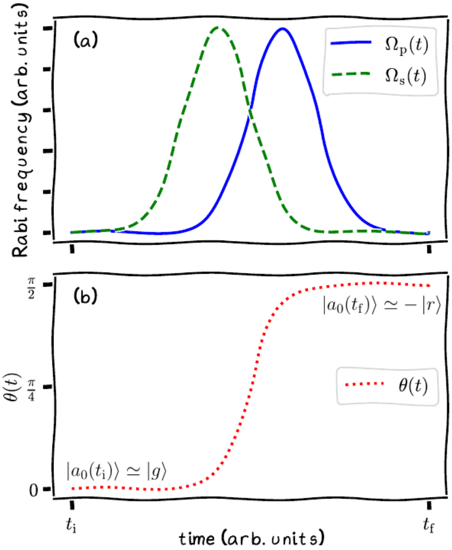
if the pulses are counterintuitively ordered

$$\lim_{t\to t_i} \frac{\Omega_p(t)}{\Omega_s(t)} = 0, \quad \lim_{t\to t_f} \frac{\Omega_s(t)}{\Omega_p(t)} = 0 \quad \Longrightarrow \quad \lim_{t\to t_i} \theta(t) = 0, \quad \lim_{t\to t_f} \theta(t) = \frac{\pi}{2}$$

then

$$|a_0(t_i)\rangle = |g\rangle \quad \text{and} \quad |a_0(t_f)\rangle = -|r\rangle$$

population transfer in a three-level system - STIRAP

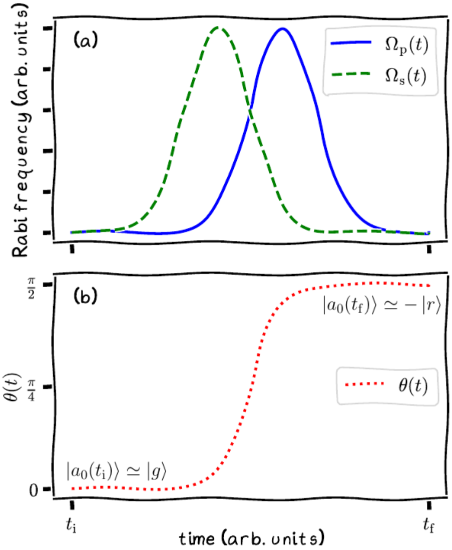


$$\lim_{t \to t_i} \frac{\Omega_p(t)}{\Omega_s(t)} = 0, \quad \lim_{t \to t_f} \frac{\Omega_s(t)}{\Omega_p(t)} = 0$$

$$\tan \theta(t) = \frac{\Omega_p(t)}{\Omega_s(t)}$$

dark state $|a_0(t)\rangle = \cos\theta(t)|g\rangle - \sin\theta(t)|r\rangle$

population transfer in a three-level system - STIRAP



$$\lim_{t\to t_i} \frac{\Omega_p(t)}{\Omega_s(t)} = 0, \quad \lim_{t\to t_f} \frac{\Omega_s(t)}{\Omega_p(t)} = 0$$

$$\tan\theta(t) = \frac{\Omega_p(t)}{\Omega_s(t)}$$

dark state $|a_0(t)\rangle = \cos\theta(t)|g\rangle - \sin\theta(t)|r\rangle$

adiabaticity condition

“area of the pulses and their overlap $\gtrsim$ 10”

population transfer in a three-level system - STIRAP

to remember

STIRAP:

- allows for efficient population transfer in a three-level system
- is characterized by the counterintuitive order of the pulses
- is an adiabatic process (area of the pulses should be *large* and they should overlap)

“super” *very short* mention of Optimal Control

“super” ***very*** short mention of Optimal Control

system described by the set of differential equations

$$\dot{\rho}(t) = f(\rho(t), \boldsymbol{u}(t), t), \quad t \in [0, T],$$

- $\rho(t)$ is the state of the system
- $\boldsymbol{u}(t) = (u_1(t), u_2(t), \dots, u_M(t))$ are controls

“super” ***very*** <u>short</u> mention of Optimal Control

system described by the set of differential equations

$$\dot{\rho}(t) = f(\rho(t), \boldsymbol{u}(t), t), \quad t \in [0, T],$$

- $\rho(t)$ is the state of the system
- $\boldsymbol{u}(t) = (u_1(t), u_2(t), \dots, u_M(t))$ are controls

introduce a cost functional whose minimization corresponds to the desired dynamics

$$\mathcal{J}(\rho(t), \boldsymbol{u}(t), T) = 1 - \mathcal{F} = 1 - \mathrm{Tr}\left\{\rho_{\mathrm{targ}}^{\dagger}\rho(T)\right\}$$

“super” *very* short mention of Optimal Control

system described by the set of differential equations

$$\dot{\rho}(t) = f(\rho(t), \boldsymbol{u}(t), t), \quad t \in [0, T],$$

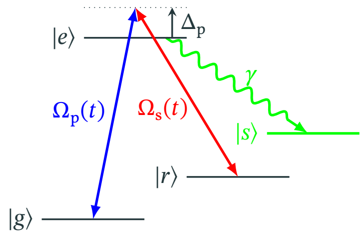
- $\rho(t)$ is the state of the system
- $\boldsymbol{u}(t) = (u_1(t), u_2(t), \dots, u_M(t))$ are controls

introduce a cost functional whose minimization corresponds to the desired dynamics

$$\mathcal{J}(\rho(t), \boldsymbol{u}(t), T) = 1 - \mathcal{F} = 1 - \mathrm{Tr}\left\{\rho_{\mathrm{targ}}^{\dagger}\rho(T)\right\}$$

$\boxed{\text{find } \boldsymbol{u}(t) \text{ which minimize } \mathcal{J}}$

Optimal Control of population transfer in three-level system



set of differential equations (master equation)

$$\dot{\rho}(t) = -\frac{\mathrm{i}}{\hbar}[H(t), \rho(t)] + \mathcal{L}_\gamma \rho(t)$$

with Hamiltonian

$$\frac{H(t)}{\hbar} = \Delta_\mathrm{p}|e\rangle\langle e| + \frac{\Omega_\mathrm{p}(t)}{2}(|\mathrm{g}\rangle\langle e| + |e\rangle\langle \mathrm{g}|) + \frac{\Omega_\mathrm{s}(t)}{2}(|e\rangle\langle r| + |r\rangle\langle e|)$$

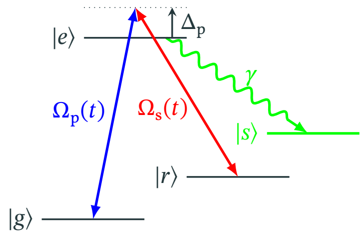
Optimal Control of population transfer in three-level system

set of differential equations (master equation)

$$\dot{\rho}(t) = -\frac{\mathrm{i}}{\hbar}[H(t), \rho(t)] + \mathcal{L}_\gamma \rho(t)$$

with Hamiltonian

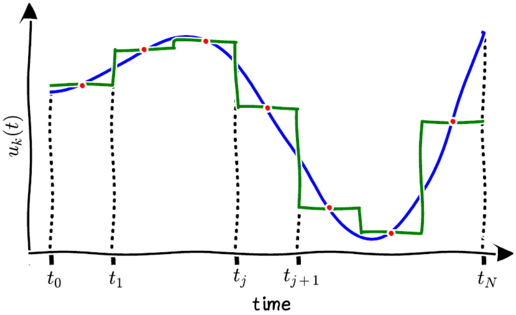
$$\frac{H(t)}{\hbar} = \Delta_\mathrm{p}|e\rangle\langle e| + \frac{\Omega_\mathrm{p}(t)}{2}(|\mathrm{g}\rangle\langle e| + |e\rangle\langle \mathrm{g}|) + \frac{\Omega_\mathrm{s}(t)}{2}(|e\rangle\langle r| + |r\rangle\langle e|)$$



the controls are $\Omega_\mathrm{p}(t)$ and $\Omega_\mathrm{s}(t)$

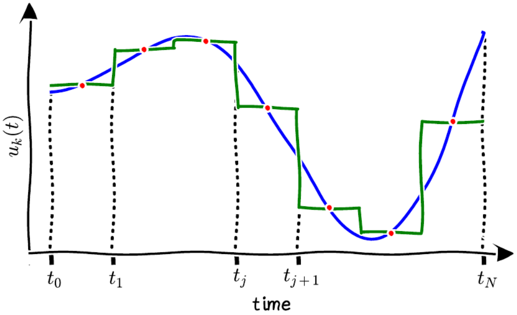
Optimal Control of population transfer in three-level system

$\Omega_\mathrm{p}(t)$ and $\Omega_\mathrm{s}(t)$ *step functions* $\longrightarrow$ each pulse parametrized by N real numbers



Optimal Control of population transfer in three-level system

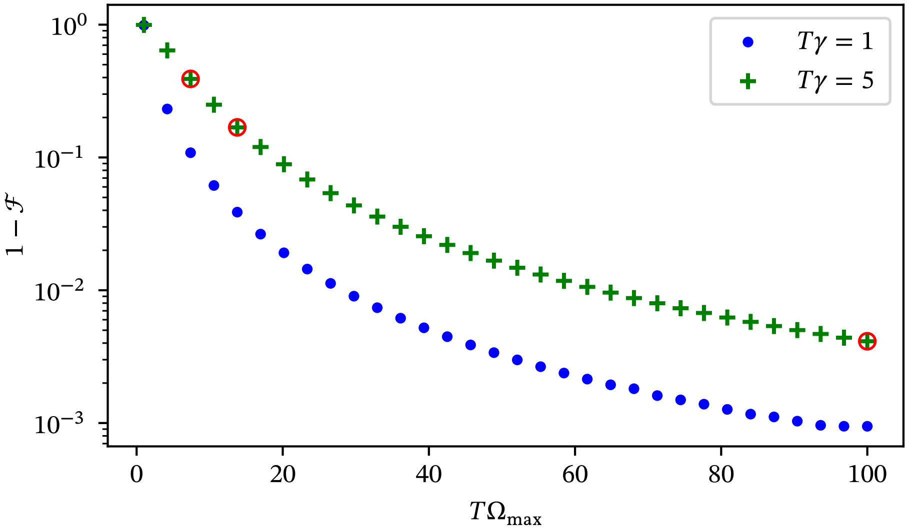
$\Omega_\mathrm{p}(t)$ and $\Omega_\mathrm{s}(t)$ *step functions* $\longrightarrow$ each pulse parametrized by N real numbers



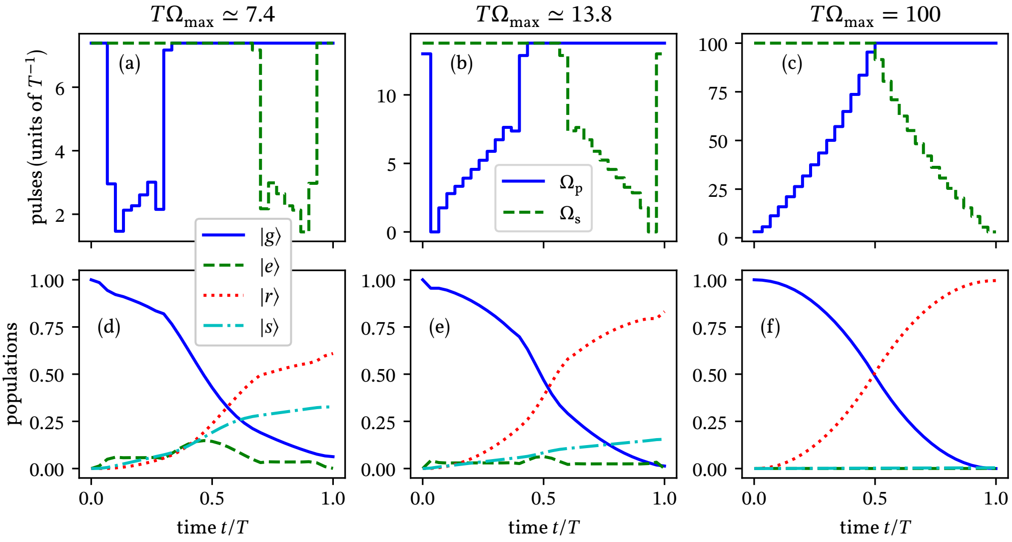
minimize the cost function $\mathcal{J}(\rho(t), \bar{\boldsymbol{u}}, T) = 1 - \mathrm{Tr}\{|r\rangle\langle r|\rho(T)\}$

with respect to $\bar{\boldsymbol{u}} = \left(\Omega_\mathrm{p}(t_0), \Omega_\mathrm{p}(t_1), \ldots, \Omega_\mathrm{p}(t_{N-1}), \Omega_\mathrm{s}(t_0), \Omega_\mathrm{s}(t_1), \ldots, \Omega_\mathrm{s}(t_{N-1})\right)$

Optimal Control of population transfer in three-level system



Optimal Control of population transfer in three-level system



"brief" idea of **Reinforcement Learning**[3]

- Markov Decision Processes (MDPs)
  - policies
  - goals, rewards and returns
  - value functions
- policy gradient and REINFORCE

[3] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, MIT press, 2018

Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs)

what are Markov Decision Processes?

Markov Decision Processes (MDPs)

what are Markov Decision Processes?

- classical formalization of sequential decision making

Markov Decision Processes (MDPs)

what are Markov Decision Processes?

- classical formalization of sequential decision making
- actions influence not just immediate rewards, but also subsequent situations

Markov Decision Processes (MDPs)

what are Markov Decision Processes?

- classical formalization of sequential decision making
- actions influence not just immediate rewards, but also subsequent situations
- mathematically idealized form of the RL problem for which precise theoretical statements can be made

Markov Decision Processes (MDPs)

what are Markov Decision Processes?

- classical formalization of sequential decision making
- actions influence not just immediate rewards, but also subsequent situations
- mathematically idealized form of the RL problem for which precise theoretical statements can be made

i will not be rigorous

Markov Decision Processes (MDPs)

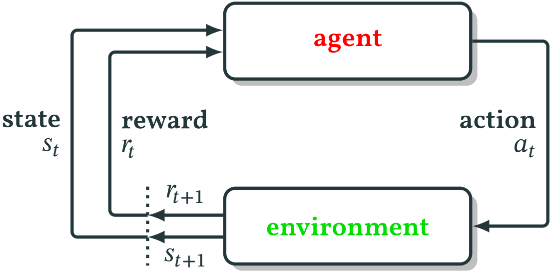
what are Markov Decision Processes?

- classical formalization of sequential decision making
- actions influence not just immediate rewards, but also subsequent situations
- mathematically idealized form of the RL problem for which precise theoretical statements can be made

i will not be rigorous

there will be some *small* imprecision

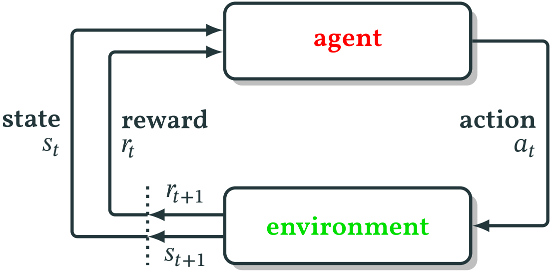
agent-environment interface



agent the learner and the decision maker

environment the thing interacting with the agent, everything outside the agent

agent-environment interface

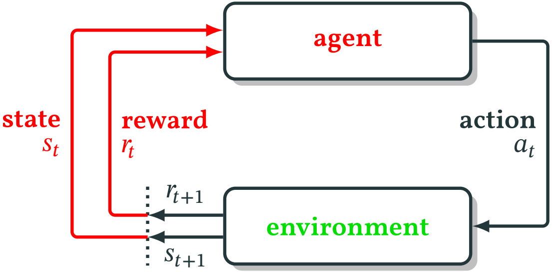


agent the learner and the decision maker

environment the thing interacting with the agent, everything outside the agent

at each discrete time step $t = 0, 1, 2, \ldots$:

agent-environment interface



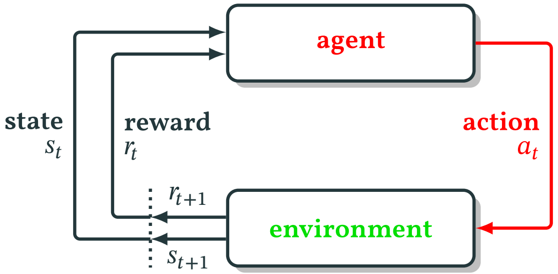
agent the learner and the decision maker

environment the thing interacting with the agent, everything outside the agent

at each discrete time step $t = 0, 1, 2, \ldots$:

- the agent receives $(r_t, s_t) = \begin{cases} s_t \in \mathcal{S} & \text{is a repr. of the environment's } \textit{state} \\ r_t \in \mathcal{R} \in \mathbb{R} & \text{is a } \textit{reward} \end{cases}$

agent-environment interface



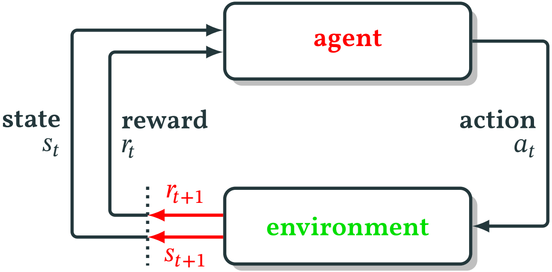
agent the learner and the decision maker

environment the thing interacting with the agent, everything outside the agent

at each discrete time step $t = 0, 1, 2, \ldots$:

- the agent receives $(r_t, s_t) = \begin{cases} s_t \in \mathcal{S} & \text{is a repr. of the environment's } \textit{state} \\ r_t \in \mathcal{R} \in \mathbb{R} & \text{is a } \textit{reward} \end{cases}$
- the agent selects an *action* $a_t \in \mathcal{A}(s_t) \longrightarrow$ environment

agent-environment interface



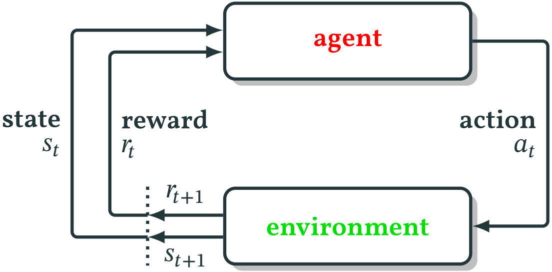
agent the learner and the decision maker

environment the thing interacting with the agent, everything outside the agent

at each discrete time step $t = 0, 1, 2, \ldots$:

- the agent receives $(r_t, s_t) = \begin{cases} s_t \in \mathcal{S} & \text{is a repr. of the environment's } \textit{state} \\ r_t \in \mathcal{R} \in \mathbb{R} & \text{is a } \textit{reward} \end{cases}$
- the agent selects an *action* $a_t \in \mathcal{A}(s_t)$ $\longrightarrow$ environment
- the environment reaches the new state s_{t+1} and produces the reward r_{t+1}

agent-environment interface



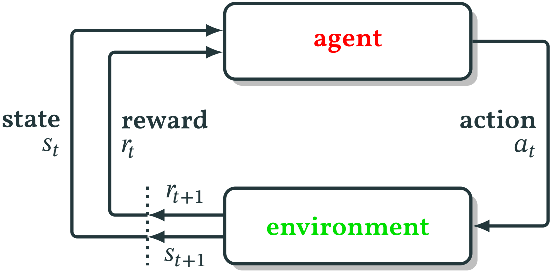
agent the learner and the decision maker

environment the thing interacting with the agent, everything outside the agent

the agent, the environment and their interaction give rise to a *trajectory*

$$\cancel{r_0 = 0}, s_0, a_0, r_1, s_1, a_1, \dots, r_t, s_t, a_t, \dots, r_N, s_N, a_N$$

agent-environment interface



agent the learner and the decision maker

environment the thing interacting with the agent, everything outside the agent

the agent, the environment and their interaction give rise to a *trajectory*

$$\cancel{r_0 = 0}, s_0, a_0, r_1, s_1, a_1, \ldots, r_t, s_t, a_t, \ldots, r_N, s_N, a_N$$

episodic task

- *termination step* $N < \infty$
- after each episode the environment is reset to an initial state

dynamics function

the environment's dynamics is completely characterized by

dynamics function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$

$$p(s', r|s, a) = \Pr\{s_t = s', r_t = r|s_{t-1} = s, a_{t-1} = a\}$$

- probability that $\begin{cases} \text{at step } t \ : & s_t = s',\ r_t = r, \\ \text{given that at step } t-1 & s_{t-1} = s,\ a_{t-1} = a \end{cases}$

dynamics function

the environment's dynamics is completely characterized by

dynamics function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$

$$p(s', r|s, a) = \Pr\{s_t = s', r_t = r|s_{t-1} = s, a_{t-1} = a\}$$

- probability that $\begin{cases} \text{at step } t : & s_t = s',\ r_t = r, \\ \text{given that at step } t-1 & s_{t-1} = s,\ a_{t-1} = a \end{cases}$

Markov property s_t and r_t depend only on s_{t-1} and a_{t-1}

dynamics function

the environment's dynamics is completely characterized by

dynamics function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0,1]$

$$p(s', r | s, a) = \Pr\{s_t = s', r_t = r | s_{t-1} = s, a_{t-1} = a\}$$

- probability that $\begin{cases} \text{at step } t : & s_t = s',\ r_t = r, \\ \text{given that at step } t-1 & s_{t-1} = s,\ a_{t-1} = a \end{cases}$

Markov property s_t and r_t depend only on s_{t-1} and a_{t-1}

not a *restriction* on the dynamics or the decision process, but a ***requirement*** on the representation of the state

MDP framework - to remember

the MDP framework is an abstraction of the problem of *goal-directed learning from interaction*

three signals passing back and forth between an agent and its environment:

- the choices made by the agent (the actions)
- the basis on which the choices are made (the states)
- definition of the agent's goal (the rewards)

MDP framework - to remember

the MDP framework is an abstraction of the problem of *goal-directed learning from interaction*

three signals passing back and forth between an agent and its environment:

- the choices made by the agent (the actions)
- the basis on which the choices are made (the states)
- definition of the agent's goal (the rewards)

formalize the task!

environment dynamics function

MDP framework - to remember

the MDP framework is an abstraction of the problem of *goal-directed learning from interaction*

three signals passing back and forth between an agent and its environment:

- the choices made by the agent (the actions)
- the basis on which the choices are made (the states)
- definition of the agent's goal (the rewards)

formalize the task!

the **actions** are what one learns

environment dynamics function

three-level population transfer as an MDP

- the environment is the three-level system and its evolution
- the agent *chooses* which pulses to apply on the system

three-level population transfer as an MDP

- the environment is the three-level system and its evolution
- the agent *chooses* which pulses to apply on the system

trajectory: $s_0, a_0, r_1, s_1, a_1, \ldots, r_i, s_i, a_i, \ldots, r_N, s_N, a_N$

three-level population transfer as an MDP

- the environment is the three-level system and its evolution
- the agent *chooses* which pulses to apply on the system

trajectory: $s_0, a_0, r_1, s_1, a_1, \ldots, r_i, s_i, a_i, \ldots, r_N, s_N, a_N$

states

$s_i = \rho(t_i) = \rho^{(i)}$

actions

$a_i = (\Omega_{\mathrm{p}}^{(i)}, \Omega_{\mathrm{s}}^{(i)})$

three-level population transfer as an MDP

- the environment is the three-level system and its evolution
- the agent *chooses* which pulses to apply on the system

trajectory: $s_0, a_0, r_1, s_1, a_1, \ldots, r_i, s_i, a_i, \ldots, r_N, s_N, a_N$

states

$s_i = \rho(t_i) = \rho^{(i)}$

actions

$a_i = (\Omega_\mathrm{p}^{(i)}, \Omega_\mathrm{s}^{(i)})$

dynamics function

evolution of the quantum system:

$$\rho^{(i+1)} = \exp\Big(\mathcal{L}(\Omega_\mathrm{p}^{(i)}, \Omega_\mathrm{s}^{(i)})\Delta t\Big)\rho^{(i)} \longrightarrow s_{i+1} = \exp[\mathcal{L}(a_i)\Delta t]s_i$$

what about r_{i+1}?

three-level population transfer as an MDP

- the environment is the three-level system and its evolution
- the agent *chooses* which pulses to apply on the system

trajectory: $s_0, a_0, r_1, s_1, a_1, \ldots, r_i, s_i, a_i, \ldots, r_N, s_N, a_N$

states

$s_i = \rho(t_i) = \rho^{(i)}$

actions

$a_i = (\Omega_{\mathrm{p}}^{(i)}, \Omega_{\mathrm{s}}^{(i)})$

dynamics function

evolution of the quantum system:

$$\rho^{(i+1)} = \exp\Big(\mathcal{L}(\Omega_{\mathrm{p}}^{(i)}, \Omega_{\mathrm{s}}^{(i)})\Delta t\Big)\rho^{(i)} \longrightarrow s_{i+1} = \exp[\mathcal{L}(a_i)\Delta t]s_i$$

what about r_{i+1}?

$$r_1 = r_2 = \cdots = r_{N-1} = 0, \quad r_N = \mathrm{Tr}\Big\{|r\rangle\langle r|\rho^{(N)}\Big\}$$

what's missing?

how the agent chooses the actions?

how the agent learns the actions that will achieve our goal?

what's missing?

how the agent chooses the actions?

how the agent learns the actions that will achieve our goal?

what is the agent?

policies

the agent chooses the actions using a policy function

a **policy** π is a mapping from *states* to *probabilities of selecting actions*

- agent following policy π at time t:
  in state s, the agent chooses action a with probability $\pi(a|s)$
- $\pi(a|s)$ is a probability distribution over $a \in \mathcal{A}(s)$ for each $s \in \mathcal{S}$

policies

the agent chooses the actions using a policy function

a **policy** π is a mapping from *states* to *probabilities of selecting actions*

- agent following policy π at time t:
  in state s, the agent chooses action a with probability $\pi(a|s)$
- $\pi(a|s)$ is a probability distribution over $a \in \mathcal{A}(s)$ for each $s \in \mathcal{S}$

Reinforcement Learning methods specify how the agent's policy is changed as a result of its interaction with the environment in order to achieve our goal

goals and rewards

reward hypothesis

all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (the reward)

goals and rewards

reward hypothesis

all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (the reward)

maximizing cumulative (weighted) sum of rewards $\longrightarrow$ achieve our goal

goals and rewards

reward hypothesis

all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (the reward)

maximizing cumulative (weighted) sum of rewards $\longrightarrow$ achieve our goal

- the reward signal is not the place to impart to the agent prior knowledge about *how* to achieve what we want it to do
- the reward signal: *what*, not *how*

goals and rewards

reward hypothesis

all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (the reward)

maximizing cumulative (weighted) sum of rewards $\longrightarrow$ achieve our goal

- the reward signal is not the place to impart to the agent prior knowledge about *how* to achieve what we want it to do
- the reward signal: *what*, not *how*

reward for three-level population transfer

$$r_1 = r_2 = \cdots = r_{N-1} = 0, \quad r_N = \mathrm{Tr}\left\{|r\rangle\langle r|\rho^{(N)}\right\}$$

return

tajectory: $s_0, a_0, r_1, s_1, a_1, \dots, r_t, s_t, a_t, \dots, a_{T-1}, r_T, s_T, \quad t = 0, 1, \dots, T$

discounted return

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots + \gamma^{T-t-1} r_T$$

- $0 \leq \gamma \leq 1$ *discount rate*,
- r_k reward at step k,
- $T \leq \infty$ *termination step*

return

tajectory: $s_0, a_0, r_1, s_1, a_1, \dots, r_t, s_t, a_t, \dots, a_{T-1}, r_T, s_T, \quad t = 0, 1, \dots, T$

discounted return

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots + \gamma^{T-t-1} r_T$$

- $0 \leq \gamma \leq 1$ *discount rate*,
- r_k reward at step k,
- $T \leq \infty$ *termination step*

at each step t the agent has to

$\boxed{\text{maximize the expected return } G_t}$

return

tajectory: $s_0, a_0, r_1, s_1, a_1, \ldots, r_t, s_t, a_t, \ldots, a_{T-1}, r_T, s_T, \quad t = 0, 1, \ldots, T$

discounted return

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots + \gamma^{T-t-1} r_T$$

- $0 \leq \gamma \leq 1$ *discount rate*, • r_k reward at step k, • $T \leq \infty$ *termination step*

at each step t the agent has to

maximize the expected return G_t

myopic agent	***farsighted*** agent
$\gamma = 0$	$\gamma \lesssim 1$

return

tajectory: $s_0, a_0, r_1, s_1, a_1, \dots, r_t, s_t, a_t, \dots, a_{T-1}, r_T, s_T, \quad t = 0, 1, \dots, T$

discounted return

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \cdots + \gamma^{T-t-1} r_T \xrightarrow{\gamma=1} G_t = r_{t+1} + r_{t+2} + r_{t+3} \cdots + r_T$$

- $0 \leq \gamma \leq 1$ *discount rate*,
- r_k reward at step k,
- $T \leq \infty$ *termination step*

at each step t the agent has to

$\boxed{\text{maximize the expected return } G_t}$

myopic **agent**	***farsighted*** **agent**
$\gamma = 0$	$\gamma \lesssim 1$

value functions

state-value function for policy π: *value of state s under policy π*

expected return when starting in s and following π thereafter

$$v_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

action-value function for policy π: *value of taking action a in state s under policy π*

expected return starting from s, taking the action a, and following policy π thereafter

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$$

value functions

state-value function for policy π: *value of state s under policy π*

expected return when starting in s and following π thereafter

$$v_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

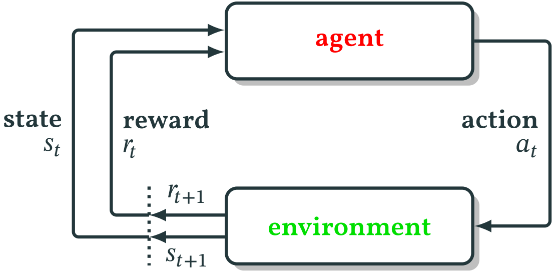
action-value function for policy π: *value of taking action a in state s under policy π*

expected return starting from s, taking the action a, and following policy π thereafter

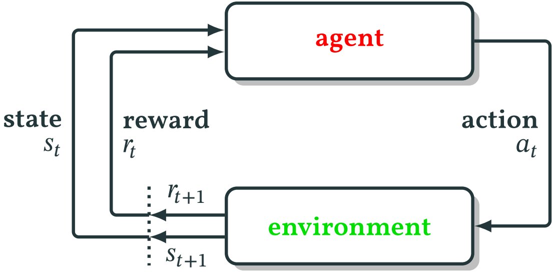
$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$$

The value functions v_π and q_π can be estimated from experience

MDP framework - to remember

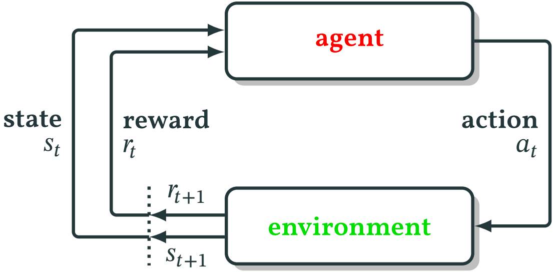


MDP framework - to remember



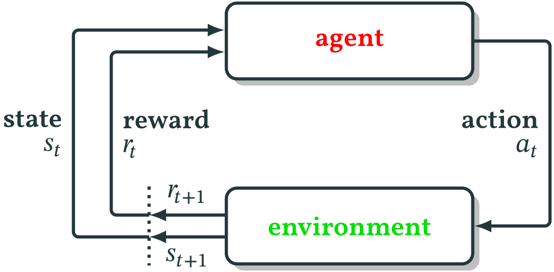
- the environment is characterized by the **dynamics function**

MDP framework - to remember



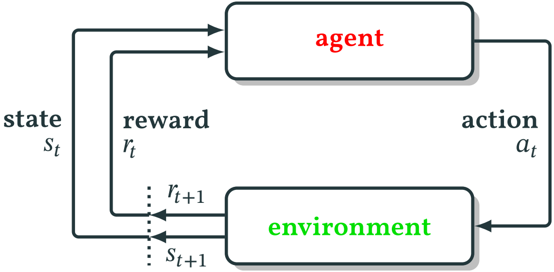
- the environment is characterized by the **dynamics function**
- the agent takes actions following a policy $\pi(a|s)$

MDP framework - to remember



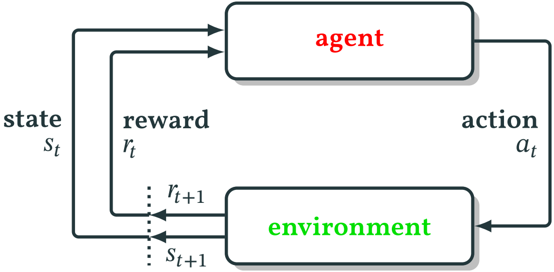
- the environment is characterized by the **dynamics function**
- the agent takes actions following a policy $\pi(a|s)$
- **Reinforcement Learning** methods specify how to change the agent 's policy in order to achieve our **goal**

MDP framework - to remember



- the environment is characterized by the **dynamics function**
- the agent takes actions following a policy $\pi(a|s)$
- **Reinforcement Learning** methods specify how to change the agent 's policy in order to achieve our **goal**
- this is done by maximizing the **expected return** G_t

MDP framework - to remember



- the environment is characterized by the **dynamics function**
- the agent takes actions following a policy $\pi(a|s)$
- **Reinforcement Learning** methods specify how to change the agent 's policy in order to achieve our **goal**
- this is done by maximizing the **expected return** G_t
- **state-value functions** and **action-value function** can be used to maximize G_t

policy gradient methods and REINFORCE

policy gradient methods

- parameterize the policy with vector $\theta \in \mathbb{R}^d$

$$\pi_\theta(a|s) = \Pr\{a_t = a | s_t = s, \theta_t = \theta\}$$

policy gradient methods

- parameterize the policy with vector $\theta \in \mathbb{R}^d$

$$\pi_\theta(a|s) = \Pr\{a_t = a | s_t = s, \theta_t = \theta\}$$

- introduce scalar performance measure $J(\theta)$

policy gradient methods

- parameterize the policy with vector $\theta \in \mathbb{R}^d$

$$\pi_\theta(a|s) = \Pr\{a_t = a | s_t = s, \theta_t = \theta\}$$

- introduce scalar performance measure $J(\theta)$
- maximize performance with approximate gradient ascent

$$\theta_{k+1} = \theta_k + \alpha \widehat{\nabla J(\theta_k)}$$

where $\widehat{\nabla J(\theta_k)} \in \mathbb{R}^d$ is a stochastic estimate of the gradient of J

policy gradient methods

- parameterize the policy with vector $\theta \in \mathbb{R}^d$

$$\pi_\theta(a|s) = \Pr\{a_t = a|s_t = s, \theta_t = \theta\}$$

- introduce scalar performance measure $J(\theta)$
- maximize performance with approximate gradient ascent

$$\theta_{k+1} = \theta_k + \alpha \widehat{\nabla J(\theta_k)}$$

where $\widehat{\nabla J(\theta_k)} \in \mathbb{R}^d$ is a stochastic estimate of the gradient of J

methods that follow this general schema are called *policy gradient* methods

REINFORCE: monte carlo policy gradient

the performance is the state-value of the initial state

$$J(\theta) = v_{\pi_\theta}(s_0)$$

and obtain the following estimate for the gradient[4]

$$\nabla J(\theta) \propto \mathbb{E}_{\pi_\theta}\left[G_t \frac{\nabla \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \right] = \mathbb{E}_{\pi_\theta}[G_t \nabla \ln \pi_\theta(a_t|s_t)]$$

REINFORCE update

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla \ln \pi_{\theta_t}(a_t|s_t)$$

[4]using the *policy gradient theorem*

REINFORCE algorithm

we have a policy π_θ parametrized by θ

REINFORCE algorithm

we have a policy π_θ parametrized by θ

1. initial guess θ

REINFORCE algorithm

we have a policy π_θ parametrized by θ

1. initial guess θ
2. generate an episode $s_0, a_0, r_1, s_1, a_1, \ldots, a_{T-1}, r_T, s_T$ following π_θ

REINFORCE algorithm

we have a policy π_θ parametrized by θ

1. initial guess θ
2. generate an episode $s_0, a_0, r_1, s_1, a_1, \dots, a_{T-1}, r_T, s_T$ following π_θ
3. for each step $t = 0, \dots, T-1$:

REINFORCE algorithm

we have a policy π_θ parametrized by θ

1. initial guess θ
2. generate an episode $s_0, a_0, r_1, s_1, a_1, \dots, a_{T-1}, r_T, s_T$ following π_θ
3. for each step $t = 0, \dots, T-1$:
    - calculate the return $G_t = r_{t+1} + r_{t+2} + \cdots + r_T$

REINFORCE algorithm

we have a policy π_θ parametrized by θ

1. initial guess θ
2. generate an episode $s_0, a_0, r_1, s_1, a_1, \ldots, a_{T-1}, r_T, s_T$ following π_θ
3. for each step $t = 0, \ldots, T-1$:
    - calculate the return $G_t = r_{t+1} + r_{t+2} + \cdots + r_T$
    - update $\theta \longleftarrow \theta + \alpha G_t \nabla \ln \pi_\theta(a_t|s_t)$

REINFORCE algorithm

we have a policy π_θ parametrized by θ

1. initial guess θ
2. generate an episode $s_0, a_0, r_1, s_1, a_1, \dots, a_{T-1}, r_T, s_T$ following π_θ
3. for each step $t = 0, \dots, T-1$:
    - calculate the return $G_t = r_{t+1} + r_{t+2} + \cdots + r_T$
    - update $\theta \longleftarrow \theta + \alpha G_t \nabla \ln \pi_\theta(a_t|s_t)$
4. if not "converged" goto 2.

REINFORCE algorithm

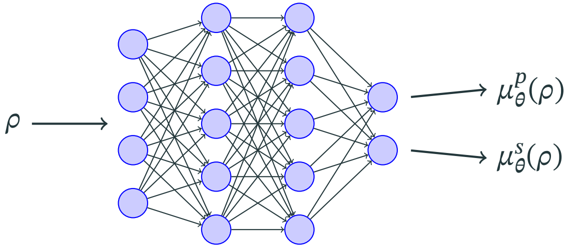
we have a policy π_θ parametrized by θ

1. initial guess θ
2. generate an episode $s_0, a_0, r_1, s_1, a_1, \dots, a_{T-1}, r_T, s_T$ following π_θ
3. for each step $t = 0, \dots, T-1$:
    - calculate the return $G_t = r_{t+1} + r_{t+2} + \cdots + r_T$
    - update $\theta \longleftarrow \theta + \alpha G_t \nabla \ln \pi_\theta(a_t|s_t)$
4. if not "converged" goto 2.

after "convergence"

agent following the policy π_θ will maximize $J(\theta) = v_{\pi_\theta}(s_0)$

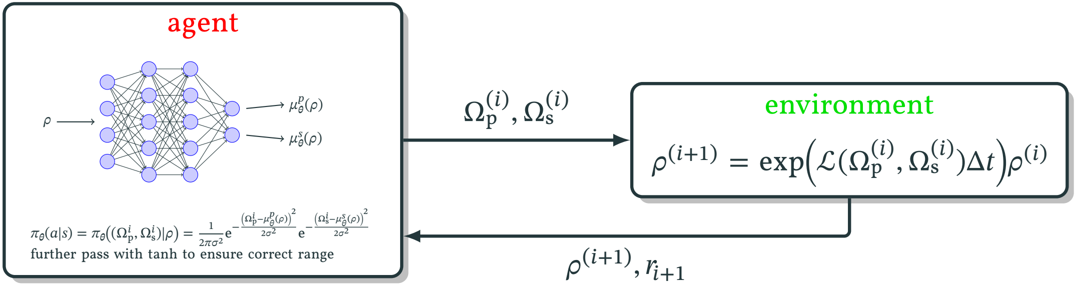
policy for population transfer in three-level systems



$$\pi_\theta(a|s) = \pi_\theta\Big((\Omega_\mathrm{p}^{(i)}, \Omega_\mathrm{s}^{(i)})|\rho\Big) = \frac{1}{2\pi\sigma^2}\mathrm{e}^{-\frac{\left(\Omega_\mathrm{p}^{(i)}-\mu_\theta^p(\rho)\right)^2}{2\sigma^2}}\,\mathrm{e}^{-\frac{\left(\Omega_\mathrm{s}^{(i)}-\mu_\theta^s(\rho)\right)^2}{2\sigma^2}}$$

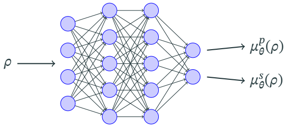
further pass with tanh to ensure correct range

policy for population transfer in three-level systems



policy for population transfer in three-level systems

agent



$$\pi_\theta(a|s) = \pi_\theta\big((\Omega_\mathrm{p}^i, \Omega_\mathrm{s}^i)|\rho\big) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{\left(\Omega_\mathrm{p}^i - \mu_\theta^p(\rho)\right)^2}{2\sigma^2}} \mathrm{e}^{-\frac{\left(\Omega_\mathrm{s}^i - \mu_\theta^s(\rho)\right)^2}{2\sigma^2}}$$

further pass with tanh to ensure correct range

$\Omega_\mathrm{p}^{(i)}, \Omega_\mathrm{s}^{(i)}$

environment

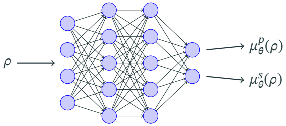
$$\rho^{(i+1)} = \exp\Big(\mathcal{L}(\Omega_\mathrm{p}^{(i)}, \Omega_\mathrm{s}^{(i)})\Delta t\Big)\rho^{(i)}$$

$\rho^{(i+1)}, r_{i+1}$

trajectory: $s_0, a_0, r_1, s_1, a_1, \ldots, r_i, s_i, a_i, \ldots, r_N, s_N, a_N$

policy for population transfer in three-level systems

agent



$$\pi_\theta(a|s) = \pi_\theta\big((\Omega_\mathrm{p}^i, \Omega_\mathrm{s}^i)|\rho\big) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{\left(\Omega_\mathrm{p}^i - \mu_\theta^p(\rho)\right)^2}{2\sigma^2}} \mathrm{e}^{-\frac{\left(\Omega_\mathrm{s}^i - \mu_\theta^s(\rho)\right)^2}{2\sigma^2}}$$

further pass with tanh to ensure correct range

$\Omega_\mathrm{p}^{(i)}, \Omega_\mathrm{s}^{(i)}$

environment

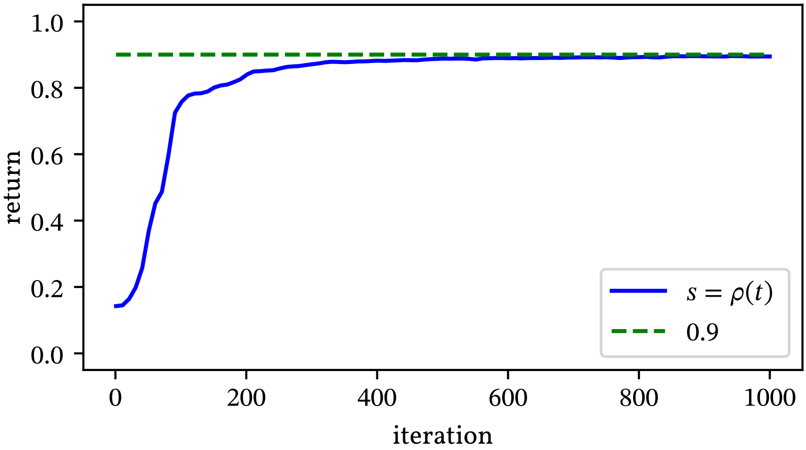
$$\rho^{(i+1)} = \exp\Big(\mathcal{L}(\Omega_\mathrm{p}^{(i)}, \Omega_\mathrm{s}^{(i)})\Delta t\Big)\rho^{(i)}$$

$\rho^{(i+1)}, r_{i+1}$

trajectory: $s_0, a_0, r_1, s_1, a_1, \ldots, r_i, s_i, a_i, \ldots, r_N, s_N, a_N$

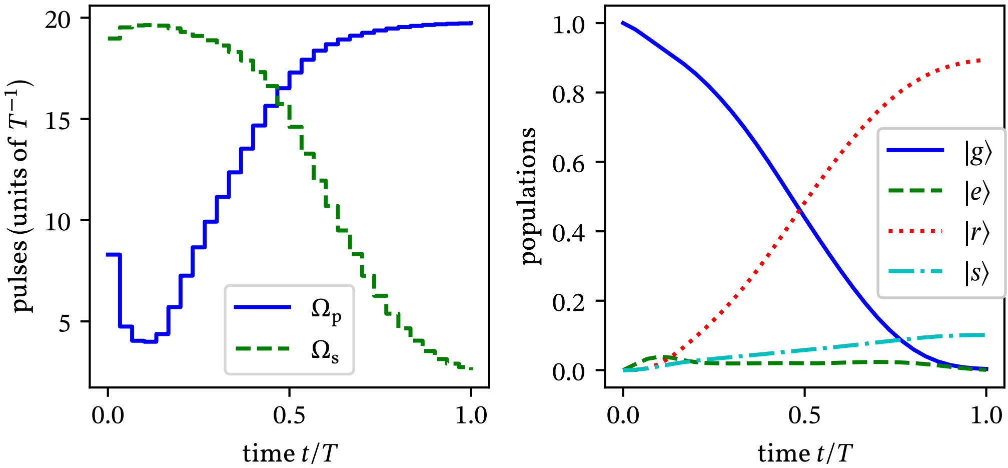
use **REINFORCE** to search for optimal policy $\pi(a, s)$

RL results



$$T\Omega_{\max} = 20, \quad T\gamma = 5$$

RL results



20
15
10
5
pulses (units of T^{-1})
0.0
0.5
1.0
time t/T
Ω_p
Ω_s
1.0
0.8
0.6
0.4
0.2
0.0
populations
0.0
0.5
1.0
time t/T
$|g\rangle$
$|e\rangle$
$|r\rangle$
$|s\rangle$

discussion and conclusions

this is not a comparison between OCT and RL

discussion and conclusions

this is not a comparison between OCT and RL

- we did not use state of the art algorithm for any of those

discussion and conclusions

this is not a comparison between OCT and RL

- we did not use state of the art algorithm for any of those
- we did not optimize the hyperparameters

discussion and conclusions

this is not a comparison between OCT and RL

- we did not use state of the art algorithm for any of those
- we did not optimize the hyperparameters
- however, in the way we implemented it:

discussion and conclusions

this is not a comparison between OCT and RL

- we did not use state of the art algorithm for any of those
- we did not optimize the hyperparameters
- however, in the way we implemented it:
  - free parameters for OCT are 60, for RL are 7644

discussion and conclusions

this is not a comparison between OCT and RL

- we did not use state of the art algorithm for any of those
- we did not optimize the hyperparameters
- however, in the way we implemented it:
  - free parameters for OCT are 60, for RL are 7644
  - computational time OCT $\approx 10^{-2}$ computational time of RL
    (but greatly varies depending on available hardware: GPUs, CPUs,..)

discussion and conclusions

this is not a comparison between OCT and RL

- we did not use state of the art algorithm for any of those
- we did not optimize the hyperparameters
- however, in the way we implemented it:
  - free parameters for OCT are 60, for RL are 7644
  - computational time OCT $\approx 10^{-2}$ computational time of RL
    (but greatly varies depending on available hardware: GPUs, CPUs,..)
  - both easily solve the problem of three-level population transfer giving STIRAP-like pulses, but OCT slightly better.

discussion and conclusions

this is not a comparison between OCT and RL

- we did not use state of the art algorithm for any of those
- we did not optimize the hyperparameters
- however, in the way we implemented it:
  - free parameters for OCT are 60, for RL are 7644
  - computational time OCT $\approx 10^{-2}$ computational time of RL
    (but greatly varies depending on available hardware: GPUs, CPUs,..)
  - both easily solve the problem of three-level population transfer giving STIRAP-like pulses, but OCT slightly better.

https://www.github.com/luigiannelli/threeLS_populationTransfer


this is the end