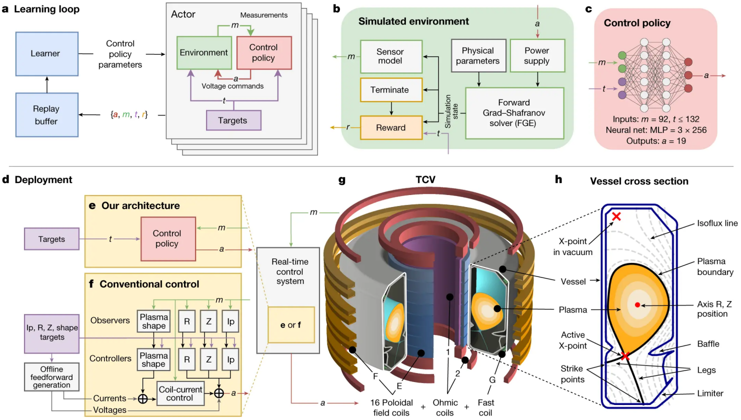
Reinforcement learning

Master CPS, Year 2 Semester 1

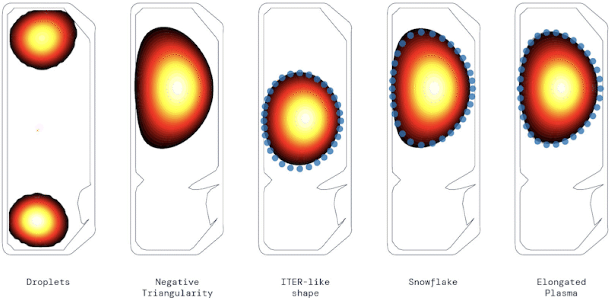
Lucian Buşoniu, Florin Gogianu

RL for plasma control in a fusion reactor (DeepMind)



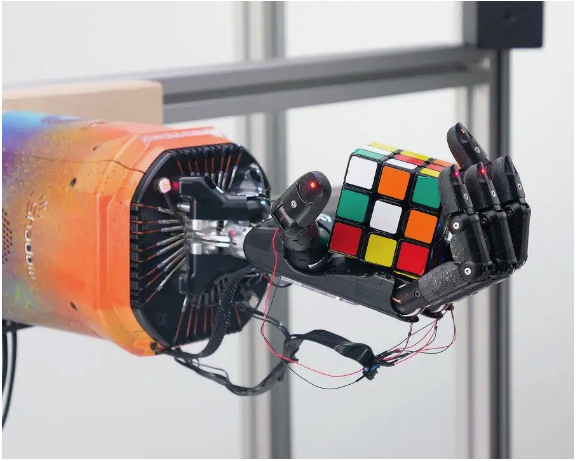
Learn to control the plasma confinement magnetic field in a simulated fusion reactor. Objective: shape and maintain high-temperature plasma.

RL for plasma control in a fusion reactor (DeepMind)



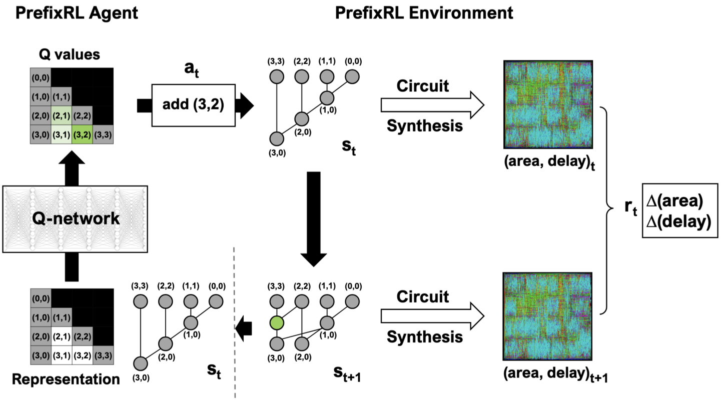
Various plasma shapes obtained by the learned controller, including a novel “droplets” configuration.

RL for manipulation of a Rubik's Cube (OpenAI)



Learn fine control of a large number of actuators, even in the presence of external disturbances.

RL design of digital circuits (Nvidia)



Learn optimal placement of parallel prefix circuits such as adders while optimising for area, delay and power.

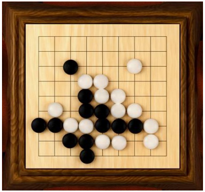
Human-level Atari with DQN (DeepMind)

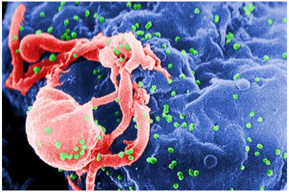
Planning for a domestic robot (UTCluj)

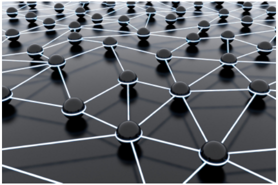
Domestic robot ensures that switches are turned off
High-level control (actions “translated” by low-level controllers into actuator commands)

Other applications

Artificial intelligence, medicine, networks of agents, economics, etc.







Course contents

- **Lecture 1: Reinforcement learning problem**
- Optimal solution
- Exact dynamic programming
- Exact reinforcement learning
- Approximation techniques
- Approximate dynamic programming
- Approximate reinforcement learning

Part I

Reinforcement learning problem

Lecture 1 contents

Why learning?

Learning finds solutions that:

1. cannot be designed in advance
   - the problem is too complex
     (e.g., control of strongly nonlinear systems)
   - the problem is incompletely known
     (e.g., robotic exploration of outer space)
2. continuously improve
3. adapt to a changing environment over time

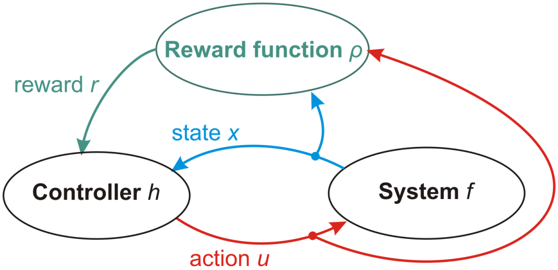
Essential for any **intelligent** system

Model-based methods

We will also focus on **model-based methods**:

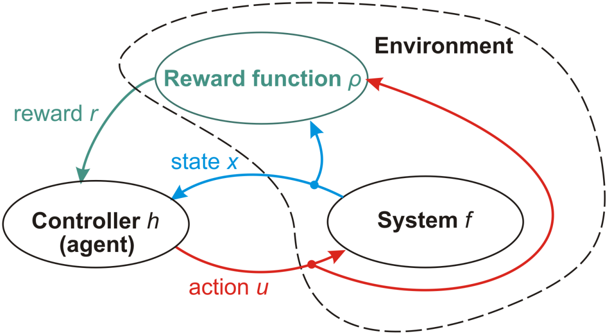
- They form the basis of reinforcement learning (e.g., dynamic programming)
- Useful independently of learning, when model available, as they address complex (e.g., nonlinear) problems

RL principle: control view



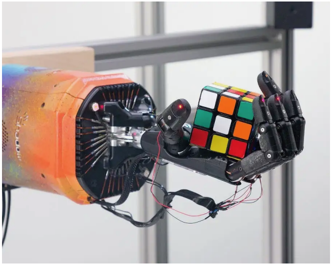
- Interact with system: measure states, apply actions
- Performance feedback in the form of rewards
- Inspired by human and animal learning

RL principle: AI view



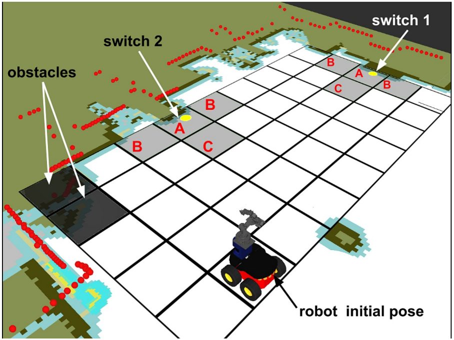
- Agent embedded in an environment
  that receives actions
  and feeds back states and rewards

Example: Rubik's cube manipulation



- States: joint angles, cube and goal positions and orientations
- Actions: 11 bins for each of the 20 actuated joints
- Rewards:
  - distance to the goal state
  - positive reward when a goal is reached
  - negative reward when a cube is dropped

Example: Domestic robot



- States: grid coordinates, switch states
- Actions: move NSEW, toggle switches
- Rewards: when a switch that was on is turned off (and penalty when an off switch is turned on!)

Example of **abstraction**: problem solved high-level, actions implemented by low-level controllers

Exact vs. approximate; deterministic vs. stochastic

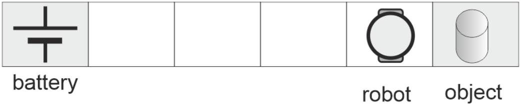
- **Parts 1–3: exact methods** – discrete states and actions with a small number of values
  - intermediate step, needed to understand the more difficult problem with approximation
  - useful on its own, if the problem can be abstracted into a high-level discrete one
- Parts 4 and onwards: approximate methods – states and actions continuous, or discrete with many values

System can behave:

- Deterministically – always responds the same to the same action in the same state
- Stochastically

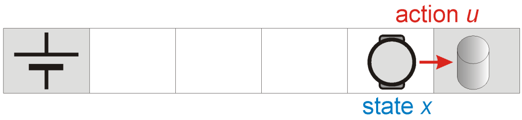
1. Introduction
2. Deterministic case
   - Markov decision process
   - Policy and objective
3. Stochastic case
4. Course organization

Simple example: cleaning robot

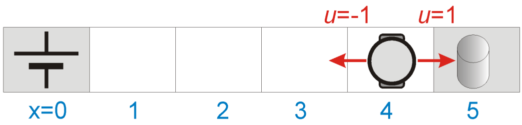


- Cleaning robot in a 1-D world
- Collects trash (reward $+5$) or battery (reward $+1$)
- After either object is collected, episode ends

Cleaning robot: state & action

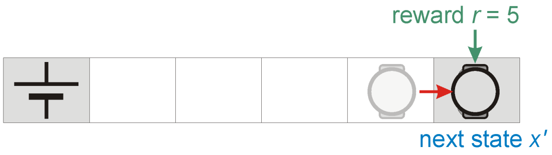


- Robot is in a state x
- and applies an action u (e.g., moves right)



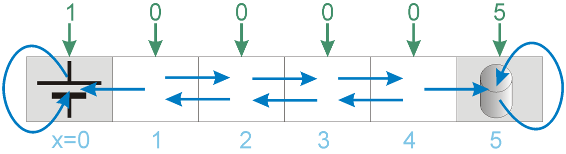
- State space $X = \{0, 1, 2, 3, 4, 5\}$
- Action space $U = \{-1, 1\} = \{\text{left}, \text{right}\}$

Cleaning robot: transition & reward



- Robot reaches a new state x'
- and receives a reward r = quality of the transition (here, $+5$ for collecting trash)

Cleaning robot: transition & reward functions



- Transition function (system behavior):

$$x' = f(x,u) = \begin{cases} x & \text{if } x \text{ terminal (0 or 5)} \\ x + u & \text{otherwise} \end{cases}$$

- Reward function (immediate performance):

$$r = \rho(x,u) = \begin{cases} 1 & \text{if } x = 1 \text{ and } u = -1 \text{ (battery)} \\ 5 & \text{if } x = 4 \text{ and } u = 1 \text{ (trash)} \\ 0 & \text{otherwise} \end{cases}$$

- **Note:** Terminal states cannot be exited and are not rewarded!

A note on rewards

- In fact, rewards depend on the **transition** $r = \tilde{\rho}(x, u, x')$
- But x' is determined by (x, u) and can be substituted in the formula:

$$\tilde{\rho}(x, u, x') = \tilde{\rho}(x, u, f(x, u)) = \rho(x, u)$$

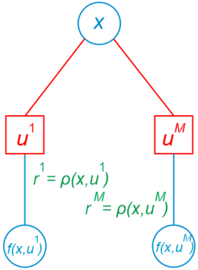
$$r = \rho(x, u) = \begin{cases} 1 & \text{if } x = 1 \text{ and } u = -1 \text{ (battery)} \\ 5 & \text{if } x = 4 \text{ and } u = 1 \text{ (trash)} \\ 0 & \text{otherwise} \end{cases}$$

Deterministic Markov decision proces

Deterministic Markov decision process

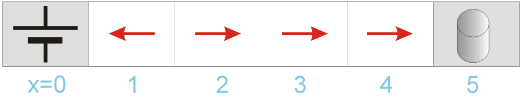
Consists of:

1. State space X
2. Action space U
3. Transition function $x' = f(x, u), \quad f : X \times U \to X$
4. Reward function $r = \rho(x, u), \quad \rho : X \times U \to \mathbb{R}$



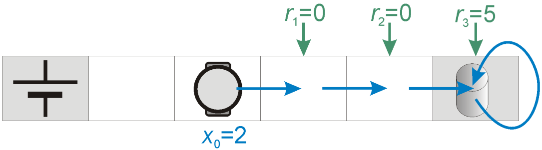
Policy

- Policy h: maps states x to actions u (state feedback)



Example: $h(0) = *$ (terminal state, action irrelevant),
$h(1) = -1$, $h(2) = 1$, $h(3) = 1$, $h(4) = 1$, $h(5) = *$

Cleaning robot: return (value)



Take h that always goes right

$$V^h(2) = \gamma^0 r_1 + \gamma^1 r_2 + \gamma^2 r_3 + \gamma^3 0 + \gamma^4 0 + \dots$$
$$= \gamma^2 \cdot 5$$

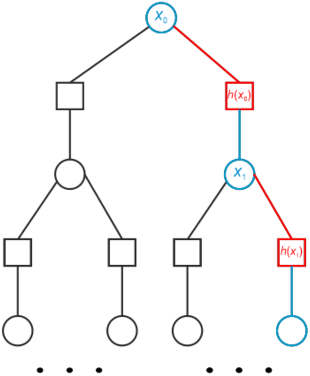
Since x_3 is terminal, all subsequent rewards are 0

General return and objective

Find h that from any x_0 maximizes the discounted return:

$$V^h(x_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, h(x_k))$$

Note: There are other types of return!



Discount factor

Discount factor $\gamma \in [0, 1)$:

- induces a "pseudo-horizon" for optimization
- bounds the infinite sum
- represents increasing uncertainty about the future
- helps algorithm convergence

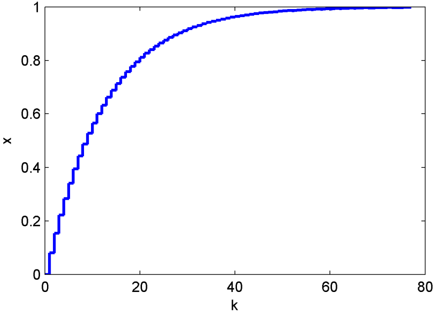
To choose γ, **trade-off** between:

1. Long-term solution quality (large γ)
2. Problem "simplicity" (small γ)

In practice, γ large enough to not ignore important rewards along system trajectories

Example: choosing γ for a first-order linear system

Step response of a first-order linear system:



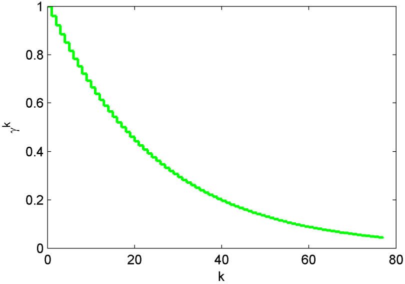
Value of γ so that rewards in steady state are visible from the initial state?

Solution: choosing γ for a first-order linear system

For $k \approx 60$, γ^k should not be too small, e.g.

$$\gamma^{60} \geq 0.05$$
$$\gamma \geq 0.05^{1/60} \approx 0.9513$$

γ^k for $\gamma = 0.96$:



Discrete random variables

- Discrete variable x can take n values, in the set $X = \{x_1, x_2, \dots, x_n\}$.
- Each value is associated with a probability $p(x_1), p(x_2), \dots, p(x_n)$, where $p(x_i) \in [0, 1]$, $\sum_i p(x_i) = 1$. $p : X \to [0, 1]$ is the **probability mass function** (PMF).

Example: The value of a die is a discrete random variable, with $n = 6$ possible values, $x_1 = 1, \dots, x_6 = 6$. For a fair die, $p(x_i) = \frac{1}{6}, \forall i = 1, \dots, 6$

Note: n can grow to infinity; mathematical description remains valid

Expected value (expectation)

- Average of the values, weighted by their probabilities; the value “expected” *a priori*, given the probability distribution:

$$\mathrm{E}\left\{x\right\} = \sum_{x \in X} p(x)x$$

Example: For a fair die, the expectation is

$$\mathrm{E}\left\{x\right\} = \frac{1}{6}1 + \frac{1}{6}2 + \ldots + \frac{1}{6}6 = 7/2$$

- A **function** with a random variable as an argument, $g : X \to \mathbb{R}$ is itself a random variable, with expectation:

$$\mathrm{E}\left\{g(x)\right\} = \sum_{x \in X} p(x)g(x)$$

Example: If faces 1-4 win 1\$, and faces 5-6 win 10\$,

$$\mathrm{E}\left\{x\right\} = \frac{1}{6}1 + \frac{1}{6}1 + \frac{1}{6}1 + \frac{1}{6}1 + \frac{1}{6}10 + \frac{1}{6}10 = 4\$$$

Independence

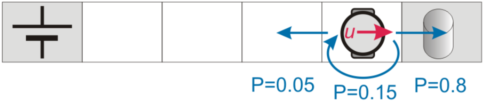
Random variables x, y are independent if the probability of vector $z = (x, y)$ is $p_z(z) = p_x(x) \cdot p_y(y)$, where p_z, p_x, p_y are the PMFs of the three variables. Note: concept extends to any number of variables

Examples:

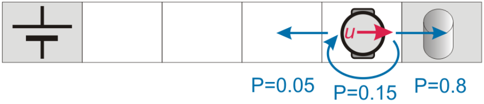
- The values of a die rolled at different times are independent. Among others, the probability of getting a 6 is independent of how many 6s were rolled in previous steps Watch out for gambler's fallacy!
- Temperature values on two consecutive days are not independent! The system is dynamic (has inertia), current values depend on previous ones

Stochastic case

- State no longer evolves deterministically, but **stochastically**
- E.g. cleaning robot “slips” and:
  - moves in the intended direction with probability (w.p.) 0.8
  - stays in place w.p. 0.15
  - moves in the opposite direction w.p. 0.05



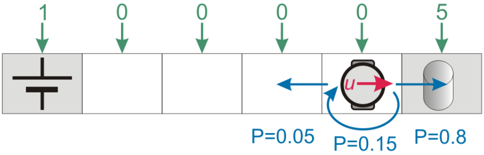
Stochastic cleaning robot: transition function



$\tilde{f}(x, u, x')$ = **probability** of reaching x' after u has been applied in x

$$\tilde{f}(x,u,x') = \begin{cases} 1 & \text{if } x \text{ terminal and } x' = x \\ 0.8 & \text{if } x \text{ non-terminal, } x' = x + u \\ 0.15 & \text{if } x \text{ non-terminal, } x' = x \\ 0.05 & \text{if } x \text{ non-terminal, } x' = x - u \\ 0 & \text{otherwise} \end{cases}$$

Stochastic cleaning robot: reward function



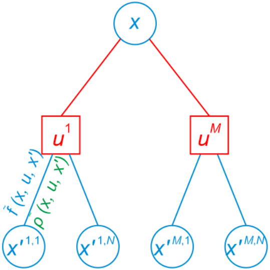
- Transition no longer fully determined by (x, u)
  $\Rightarrow$ the next state x' must be explicitly included
- $\tilde{\rho}(x, u, x')$ = reward on reaching x'
  as a result of action u in x
- For cleaning robot:

$$\tilde{\rho}(x, u, x') = \begin{cases} 5 & \text{if } x \neq 5 \text{ and } x' = 5 \\ 1 & \text{if } x \neq 0 \text{ and } x' = 0 \\ 0 & \text{otherwise} \end{cases}$$

Stochastic Markov decision process

Stochastic Markov decision process

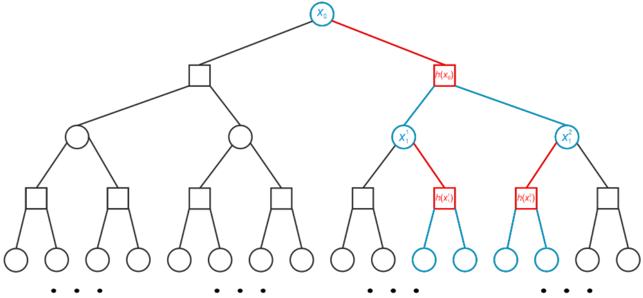
1. State space X
2. Action space U
3. Transition function $\tilde{f}(x, u, x')$, $\quad \tilde{f} : X \times U \times X \to [0, 1]$
4. Reward function $\tilde{\rho}(x, u, x')$, $\quad \tilde{\rho} : X \times U \times X \to \mathbb{R}$



Objective in stochastic case

Find h that from any x_0 maximizes expected discounted return:

$$V^h(x_0) = \mathrm{E}_{x_1, x_2, \dots} \left\{ \sum_{k=0}^{\infty} \gamma^k \tilde{\rho}(x_k, h(x_k), x_{k+1}) \right\}$$

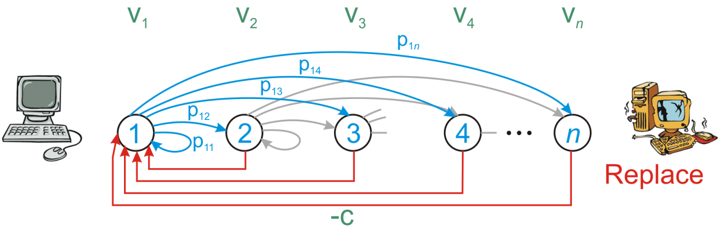


Policy, discount in stochastic case

- Policy $h(x)$ has the same structure,
- discount factor γ has the same meaning

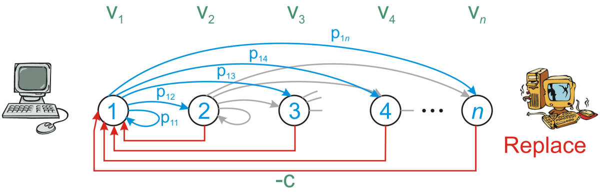
as in the deterministic case

Example: machine replacement



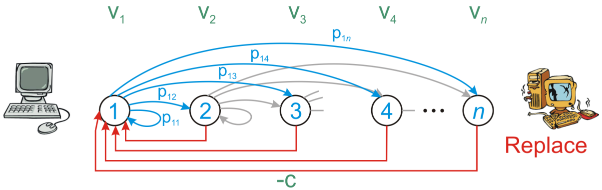
- Machine with *n* different states = wear levels
  1=pristine, *n*=fully degraded
- Produces revenue v_i operating in state i
- Stochastic wear: wear level i transitions to $j > i$ w.p. p_{ij}, remains i w.p. $p_{ii} = 1 - p_{i,i+1} - \dots - p_{i,n}$
- Machine can be instantaneously replaced at any time, paying cost c

Machine replacement: State and action spaces



- State space $X = \{1, 2, \ldots, n\}$
- Action space $U = \{\text{Wait}, \text{Replace}\}$

Machine replacement: Transition and reward functions



- Transition function:

$$\tilde{f}(x = i, u, x' = j) = \begin{cases} p_{ij} & \text{if } u = \text{W and } i \leq j \\ 1 & \text{if } u = \text{R and } j = 1 \\ 0 & \text{in any other situation} \end{cases}$$

- Reward function:

$$\tilde{\rho}(x = i, u, x' = j) = \begin{cases} v_i & \text{if } u = \text{W} \\ -c + v_1 & \text{if } u = \text{R} \end{cases}$$

Machine replacement: motivation

The RL framework provides a way to formalize and find an optimal decision policy that maximizes the long-term value of the machine

$$V^h(x_0) = \mathrm{E}_{x_1, x_2, \ldots} \left\{ \sum_{k=0}^{\infty} \gamma^k \tilde{\rho}(x_k, h(x_k), x_{k+1}) \right\}$$

Key terms in this lecture

- reinforcement learning, RL
- state
- action
- reward
- transition function
- reward function
- Markov decision process
- policy
- return
- discount factor
- random variable
- probability mass function
- expected value

Bibliography

Mandatory material: course slides

Optional books:

- R. Sutton, A. Barto, Reinforcement Learning: An Introduction, ed. 2, 2018.
- D. Bertsekas, Dynamic Programming and Optimal Control, vol. 2, Athena Scientific, 2012.
- D. Bertsekas, Reinforcement Learning and Optimal Control, Athena Scientific, 2024.
- L. Buşoniu, Reinforcement learning and dynamic programming for control, 2012 (lecture notes).

Logistics

Grading:

- 50% labs
- 50% exam
- 10% lecture quizzes

Lab rules:

- labs **mandatory before joining the exam**
- solution = PDF report + code: max 10p if submitted on time, max 5p if late
- solutions must be validated through discussions
- any copied or LLM-generated lab $\Rightarrow$ ineligible and re-enroll

Website, contact

`http://busoniu.net/teaching/rl2024`

Email: `lucian@busoniu.net`, `florin.gogianu@gmail.com`

Info

- Course lectures (slides)
- Labs
- Schedule
- etc.

Quiz

Quiz