

Written examination date: May 27th, 2024

Course title: Introduction to reinforcement learning and control

Course number: 02465

Aids allowed: All aids allowed

Exam duration: 4 hours

Weighting: The exam is divided into 3 parts:

- Multiple-Choice questions
- Conceptual questions
- Programming questions

The overall scores of each part contribute roughly equally towards the overall result. Each question in each part contribute equally towards the score of that part.

Part I: Questions 1-9 are multiple choice. The score of a correct answer is 3 points. The score of an incorrect answer is -1 points. The score of option E or blank is 0 points.

Part II and part III: Each completed sub-task contribute towards your score.

Preparing your handin: The three parts are prepared as follows:

Part I: Edit the file `irlc/exam/exam2024spring/multiple_choice_answers.py`. Don't include calculations. Only answer with `'A'`, `'B'`, `'C'`, `'D'`, `'E'`.

Part II: Create a PDF file with your answers and justifications.

Part III: Program your answer in the `.py`-files indicated in the question and run `irlc/exam/exam2024spring/exam2024spring_tests_grade.py` to generate your `.token`-file.

Handing in: To hand in, you should upload the files:

- The `irlc/exam/exam2024spring/multiple_choice_answers.py`-file with your answer to part I
- The `.pdf`-file with your answers to part II
- The `.token`-file with your answers to part III
- The `irlc/exam/exam2024spring/question_inventory.py` source file containing your solution
- The `irlc/exam/exam2024spring/question_bill_mdp.py` source file containing your solution
- The `irlc/exam/exam2024spring/question_control.py` source file containing your solution

Note on part II: The main quantities asked for are highlighted as $f(x)$. Answer unambiguously, concisely, and if applicable with algebraic simplifications. Your final result must be clearly indicated at the end of your answer: $f(x) = 3 \sin(x)$. To get credit, you must state the relevant theory and equations, and all relevant calculations must be included. Credit is not given for answers with missing or erroneous justifications.

Note on part III: To get started, move the folder `irlc/exam/exam2024spring` from the `.zip` file to the corresponding location in your existing exercise directory. The `.py` source files must be **reproducible** and **readable** so that someone else can run and fully understand your solution. You can freely use the `irlc`-toolbox (including solutions) and the packages we have used in the course, but you **must** include additional code you write during the exam or have prepared beforehand in the source files listed above. The source files must not be renamed. The `.token` file contains your results and must be up to date with your source files, i.e., generate it just prior to handin. In the case they differ, the `.token` file takes precedence. Credit is given for correct implementations defined by the problem description. The points in the `.token` file name is computed from the public tests, and might not correspond to overall correctness.

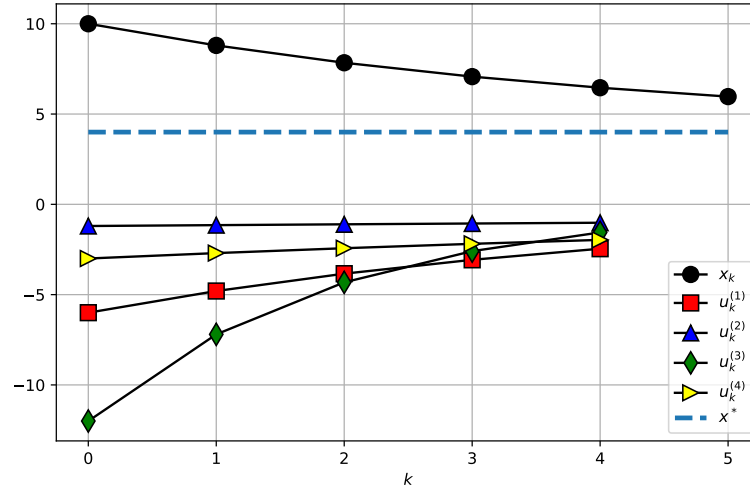


Figure 1: Plot the behavior of a system x_k and four possible PID control trajectories $u_k^{(1)}$, $u_k^{(2)}$, $u_k^{(3)}$, and $u_k^{(4)}$.

Part I: Multiple-Choice

Question 1:

Suppose we wish to steer a servo motor towards a desired angle $x^* = 4$. This can be accomplished by applying a control signal u_k at time steps $k = 0, 1, \dots$, computed from the current angle x_k using a PID controller with target x^* and PID constants:

$$K_p = 2.0, \quad K_i = 0.0, \quad K_d = 0.0.$$

Figure 1 shows the behavior of both x_k , as well as four possible control signals $u_k^{(1)}$, $u_k^{(2)}$, $u_k^{(3)}$, and $u_k^{(4)}$ at time steps $k = 0, 1, 2, \dots$.

Which one of the four possible control signals corresponds to the true control signal u_k of the system?

- A. Control trajectory $u_k^{(1)}$
- B. Control trajectory $u_k^{(2)}$
- C. Control trajectory $u_k^{(3)}$
- D. Control trajectory $u_k^{(4)}$
- E. Don't know.

Question 2:

Consider an episodic MDP, and assume as usual that G_t represents the γ -discounted accumulated return. Which one of the following (alternative) Bellman-like recursions are true for the value function v_π ?

- A. $v_\pi(s) = \frac{1}{2} \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) + G_t \mid S_t = s]$
- B. $v_\pi(s) = \frac{1}{2} \mathbb{E}[G_t + \gamma v_\pi(S_{t+1}) \mid S_t = s]$
- C. $v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma G_t \mid S_t = s]$
- D. $v_\pi(s) = \mathbb{E}\left[\frac{R_{t+1} + R_{t+2}}{2} + \gamma v_\pi(S_{t+2}) \mid S_t = s\right]$
- E. Don't know.

Question 3:

Which one of the following options are correct about iterative LQR as seen in the course?

- A. The quantity α denotes the learning rate. It represents a tradeoff between stability and rate of convergence, and is selected by the user.
- B. It use an optimizer (such as scipy) to determine the optimal actions
- C. At a given time step k , the policy computed by iterative LQR is a linear function of the input state \mathbf{x}_k
- D. It uses Model-predictive control to plan on a short horizon
- E. Don't know.

Question 4:

Which one of the following statements about Every-visit Monte-Carlo prediction for estimating the value function v_π is correct?

- A. Every-visit Monte-Carlo prediction can be used to estimate the value function v_π when the policy π is deterministic.
- B. The prefix *every* in every-visit Monte-Carlo prediction refers to the fact every *action* is tried at least once in every episode
- C. The prefix *every* in every-visit Monte-Carlo prediction refers to the fact every *state* is visited at least once in every episode
- D. Every-visit Monte-Carlo prediction is biased in the sense that during training, it will tend to systematically over-estimate the value function
- E. Don't know.

Question 5:

Which one of the following statements are true about tabular double Q -learning ([SB18, Chapter 6])?

- A. Double- Q learning reduces bias in the estimated action-value function by computing the Q -values as the average of two biased estimates Q_1 and Q_2
- B. Double- Q learning reduces bias by letting Q_1 under-estimate the optimal action-value function and letting Q_2 over-estimate the optimal action-value function
- C. In each step t , either Q_1 or Q_2 are updated by double- Q learning. If both were updated, the algorithm would estimate $2q_*$ and not q_* (i.e. twice the optimal action-values)
- D. After running the method for a long time the two Q functions will be approximately equal $Q_1 \approx Q_2$
- E. Don't know.

Question 6:

Consider the DP algorithm applied to the inventory control example described in [Her25, section 5.1.2]. Assume that we plan on a long horizon N (for instance $N = 1000$), and consider k as corresponding to being approximately halfway through the planning, i.e. $k \approx \frac{N}{2}$. Which one of the following statements are true?

- A. The function J_k requires about $\frac{N}{2}$ times more memory to store than J_1
- B. $J_0(x_k) < J_k(x_k)$
- C. $J_k(x_k)$ represents the expected accumulated cost the optimal policy obtained during planning rounds $0, 1, \dots, k$
- D. $J_k(x_k) \geq 0$
- E. Don't know.

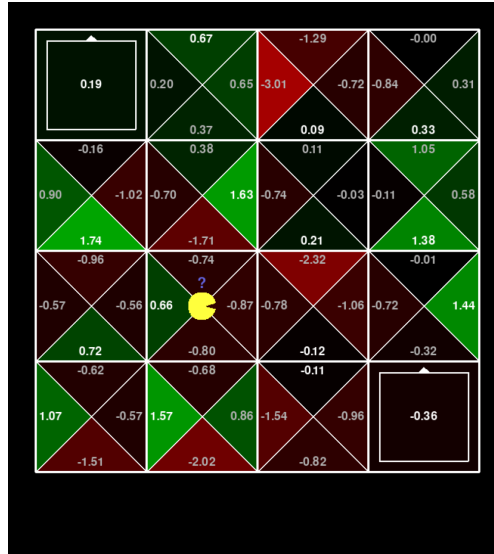


Figure 2: A gridworld environment with randomly initialized Q -values. The living reward is $+1$.

Question 7:

We consider the familiar gridworld environment discussed in [Her25, section 4.2.4] shown in fig. 2.

Suppose that Q -learning (using discount factor $\gamma = 0.9$ and learning rate $\alpha = 0.8$) is applied to the Gridworld MDP shown in fig. 2. The living reward is $R_t = 1$, and the dynamics is deterministic. The current state s_t of the agent is as indicated in the figure.

As can be seen, the Q -values have been initialized randomly, but this does not alter how Q -learning function. Assuming the agent takes the action **North**, what will be the new value of the Q -value indicated by the blue question mark?

- A. 1.826
- B. 1.957
- C. 2.468
- D. 2.567
- E. Don't know.

Question 8:

Which of the following statements are true about UCB and the 10-armed testbed described in [SB18, Chapter 2]?

- A. Assuming the average rewards $q^*(a)$ are fixed across runs, it will still be the case that the action sequence of actions a_1, a_2, \dots generated by UCB will be different across different runs.
- B. Over time, the upper-confidence bounds which are used to select actions, $Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}$, will all converge to the true Q -values $q_*(a)$
- C. The number $c \geq 0$ acts as a regularization constant. A higher value of c will tend to stabilize the algorithm numerically, at the cost that the $Q_t(a)$ -estimates seen during training will be driven towards 0 in proportion to c .
- D. In the 10-armed testbed, if c is very large (i.e., $c > 10^6$), UCB may diverge
- E. Don't know.

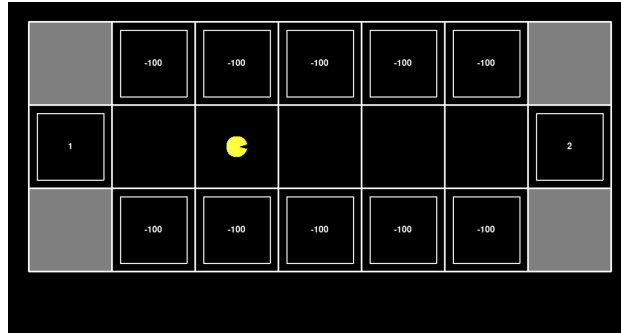


Figure 3: An example gridworld environment. The location of Pacman indicates the initial state.

Question 9:

Consider the optimal control by direct collocation method applied to the Pendulum environment (in the

$\mathbf{x} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$ parameterization). It is assumed the cost-function has the form:

$$\int_0^{t_F} \left[1 + \left(\mathbf{x}(\tau) - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right)^\top \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \left(\mathbf{x}(\tau) - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \right] d\tau$$

Given an arbitrary initial position, which one of the following statements are true?

- A. The control method will try to minimize the control signal $u(t)$
- B. The control method will try to bring the system to the position $\theta = 0$.
- C. The control method will try to bring the system to a standstill $\dot{\theta} = 0$.
- D. The control method will try to minimize the time spend interacting with the system t_F
- E. Don't know.

Part II: Conceptual questions

Question 10:

Suppose that Monte-Carlo prediction (using discount factor $\gamma = 0.8$) is applied to the Gridworld MDP shown in fig. 3. The living reward is 0, and the dynamics is deterministic. The gridworld obey the usual rules, meaning that the square boxes with numbers indicate exit squares, so that if Pacman lands in one of them, only one action $a_t = 0$ (**North**) is available upon which the environment immediately terminates with the reward given in the square.

- (a) Suppose we apply first-visit Monte carlo prediction (see [SB18, Section 5.1]) to the problem and the first episode consists of first going **West**, and then moving **East** to the goal, resulting in the actions:

$$a_0 = \text{West}, a_1 = \text{East}, a_2 = \text{East}, a_3 = \text{East}, a_4 = \text{East}, a_5 = \text{East}, a_6 = \text{North}.$$

Assume the value function is initialized to zero. What is the updated value function at the starting state, $V(s_0)$?

- (b) Consider the same list of actions as in the previous questions, but assume that instead of first-visit Monte carlo we had used every-visit Monte carlo instead. In this case, what is the value function in the initial state $V(s_0)$?

Question 11:

Consider a control problem where a control signal $u(t) \in \mathbb{R}$ is applied to control a system with state $x(t) \in \mathbb{R}$ and where the dynamics satisfy the following differential equation:

$$\dot{x}(t) = 2 \sin(t + x(t) + u(t)), \quad t \geq 0. \quad (1)$$

In the following, we suppose that problem is Euler discretized using a time constant of $\Delta = 0.25$ to yield states x_0, x_1, x_2, \dots and control signals u_0, u_1, u_2, \dots , each corresponding to the state of the system at time step t_k , starting at $t_0 = 0$.

- Assume that $x_0 = 0$, and a constant control signal of $u_k = -\Delta$ is applied to the system. What is the numerical value of x_1 ?
- Ignore the previous question and assume the control signal follows the rule $u_k = -2x_k - \Delta k$. When evaluating the derivative term in PID control, we need to compute the discrete approximation of the derivative defined as: $\frac{x_{k+1} - x_k}{\Delta}$. Simplify this expression so that it *only* depends on x_k .
- Ignore the previous two questions and consider the continuous-time formulation of the system defined in eq. (1). Suppose the system is initialized in $x(0) = 0$, and that we then apply a closed-loop control signal $u(t) = -x(t)$ to the system. Derive a simple, closed-form expression for the control signal $u(t)$, and use it to compute the numerical value of $u(\frac{\pi}{2})$.

Question 12:

Consider a finite-horizon dynamical programming problem where the terminal cost is

$$g_N(x_N) = e^{x_N} - 1. \quad (2)$$

- Suppose that the dynamics is $f_k(x_k, u_k, w_k) = x_k + u_k + w_k$, and that when Dynamical Programming is applied to this problem, it gives rise to the following update rule for J_N :

$$J_{N-1}(x_{N-1}) = \min_{u \in \mathcal{A}_{N-1}(x_{N-1})} \mathbb{E}_{w_{N-1}} [3u + 2 + e^{x_{N-1} + u + w_{N-1}}].$$

Use this information to determine g_{N-1} .

- Ignore the previous question and instead assume that, in addition to eq. (2), you are told that for $k = 0, \dots, N-1$:

$$\begin{aligned} f_k(x_k, u_k, w_k) &= x_k + u_k + w_k, \\ g_k(x_k, u_k, w_k) &= x_k, \\ p_k(w_k | x_k, u_k) &= \frac{e}{e-1} e^{-w_k}, \quad \text{and} \quad 0 \leq w_k \leq 1 \\ \mathcal{A}_k(x_k) &= \{0, 1\}, \end{aligned}$$

Use this information¹ to compute the numerical value of $J_{N-1}(2)$.

- Assume f_k , g_k and w_k are as described in the previous question. Assume that the state space at $k = N-1$ (i.e., the states the system can be in at planning step $N-1$) is comprised of just the unit interval:

$$\mathcal{S}_{N-1} = [0; 1].$$

What are the possible states the system can be in at the next time step N ? I.e., what is the smallest possible state space, \mathcal{S}_N , compatible with this information?

¹The density of the random noise terms w_k is normalized since $\int_0^1 e^{-w} dw = 1 - \frac{1}{e}$.

Part III: Programming questions

Question 13:

To solve this question, you should edit the file `irlc/exam/exam2024spring/question_inventory.py`. This problem is about a variant of the inventory control problem discussed in [Her25, section 5.1.2]. We consider an inventory control problem that represents a store which sells bridal gowns such that x_k denotes the number of bridal gowns in stock at planning round k . The original inventory control model and the dynamical programming algorithm is included in the exam folder.

The bridal gown store problem is equivalent to the inventory control problem on a horizon of N with two changes:

- There are m actions, $\mathcal{A}_k(x_k) = \{0, 1, \dots, m-1\}$.
- Customers can buy from $w = 0$ to $w = m-1$ bridal gowns, and the distribution of the number of bridal gowns customers buy each day is:

$$p_W(w_k = i \mid x_k, u_k) = \frac{1}{m}, \quad i = 0, \dots, m-1.$$

Note that the state spaces, cost functions, and the dynamics are the same as in the inventory control problem.

- Complete `def a_get_cost(N: int, m: int, x0: int)`: This function is given a value of N , m and a starting state x_0 . It should then compute the optimal cost when starting in state x_0 and planning on a horizon of N , i.e. $J^*(x_0)$, as a `float`.
- Complete `def b_sale(N: int, m: int, x0: int)`: Suppose that in each planning round, the store owner has an additional action $u_k = \text{sale}$ corresponding to holding a sale. When the store owner holds a sale, she orders a number of gowns, sell them at a reduces price, and is guaranteed to end up with an empty inventory. Concretely this implies two changes:

$$x_{k+1} = f_k(x_k, u_k, w_k) = \begin{cases} 0 & \text{If there is a sale} \\ \text{usual value} & \text{otherwise} \end{cases}, \quad g_k(x_k, u_k, w_k) = \begin{cases} \frac{3}{4}(m - w_k) & \text{If there is a sale} \\ \text{usual value} & \text{otherwise.} \end{cases}$$

As in the previous problem, the function should accept N , m , and x_0 as inputs and return expected cost $J^*(x_0)$ as a `float`. *Hint: Account for the sale-action as an extra element in the action space.*

Question 14:

To solve this question, you should edit the file `irlc/exam/exam2024spring/question_bill_mdp.py`. Bill is renting an apartment, and is now looking into how he can best make decisions to maximize his profit in each subsequent year $t = 0, 1, 2, \dots$.

This decision problem is formulated as a MDP, where Bills objective is the usual MDP objective of maximizing the expected γ -discounted return (we assume that $0 < \gamma < 1$). Specifically, the MDP is defined as follows:

- There are two states and the problem never terminates.
 - One state corresponds to Bill owning the apartment. There are then two actions available to him:
 - He can move in with his parents and AirBnB his apartment. This is guaranteed to result in a reward of $R_{t+1} = r_{\text{airbnb}}$ (for instance $r_{\text{airbnb}} = 0.01$)
 - He can gamble the lease to the apartment at the casino. With probability $p_w = 0.45$, he will gain two units of money $R_{t+1} = 2$, and with probability $1 - p_w$, he will permanently loose the apartment.
 - The other state corresponds to Bill having lost his apartment.
 - Bill now lives permanently with his parents and will always get a reward of $R_{t+1} = 0$.
 - Bill starts in the state corresponding to owning the apartment, s_0
- (a) Complete `def a_always_airbnb(r_airbnb : float, gamma : float)`: Assume Bill play economically safe and always decide to move in with his parents. In each round he therefore always get an AirBnb income of r_{airbnb} (corresponding to `r_airbnb`). For a given discount factor γ (corresponding to `gamma`), the function should compute the expected future γ -discounted return $v_\pi(s_0)$ when Bill follows this policy. The function should return a single number.
- (b) Complete `def b_random_decisions(r_airbnb : float, gamma : float)`: Bill is unhappy with living with his parents and acts a bit erratic. Bill therefore follows the random policy π where, as long as he owns the apartment, he will gamble with a probability of 0.5 and otherwise AirBnb the apartment. The function should compute $v_\pi(s_0)$ when Bill follows this policy for different values of γ and r_{airbnb} . The function should return a single number.
- (c) Complete `def c_is_it_better_to_gamble(r_airbnb : float, gamma : float)`: This function should return `True` when the optimal policy, computed using the given value of r_{airbnb} and γ , dictates that Bill should gamble and otherwise `False`.

Question 15:

To solve this question, you should edit the file `irlc/exam/exam2024spring/question_control.py`. In this problem, we will consider a control task with one-dimensional state and action vectors, $x(t), u(t) \in \mathbb{R}$. The dynamics obey the 1-d differential equation:

$$\dot{x} = f(x, u) = -e^{u-x^2}.$$

and there are no constraints on the system.

- (a) Complete `def a_xdot(x : float, a : float)`: Assume that a closed loop policy of the form $u(t) = ax(t)^2$ is applied to the system. Given the state x and a value for a , the function should return the rate of change in the state \dot{x} as a `float`.
- (b) Complete `def b_rk4_simulate(u0 : float, tF : float)`: Suppose that the system is initialized in $x(0) = 0$ and a constant policy $u(t) = u_0$ (corresponding to `u0`) is applied to the system until a time t_F (corresponding to `tF`). The function should return the end-state $x(t_F)$, represented as a `float`, computed using RK4 integration as recommended in the course material.

References

[Her25] Tue Herlau. Sequential decision making. (Freely available online), 2025.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. (Freely available online).

This line concludes the exam. Document build: 2025/02/09, 15:47:47.