Written examination date: May 24th, 2023
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/exam2023spring/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/exam2023spring/exam2023spring_tests_grade.py` to generate your `.token` -file.
Handing in: To hand in, you should upload the files:

- The `irlc/exam/exam2023spring/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/exam2023spring/question_inventory.py` source file containing your solution

- The `irlc/exam/exam2023spring/question_lqr.py` source file containing your solution

- The `irlc/exam/exam2023spring/question_bandit.py` source file containing your solution

Note on part II: The main quantities asked for are highlighted as $\boxed{f(x)}$. Answer unambigiously, concisely, and if applicable with algebraic simplifications. Your final result must be clearly indicated at the end of your answer: $\boxed{f(x) = 3\sin(x)}$. 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/exam2023spring` 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.
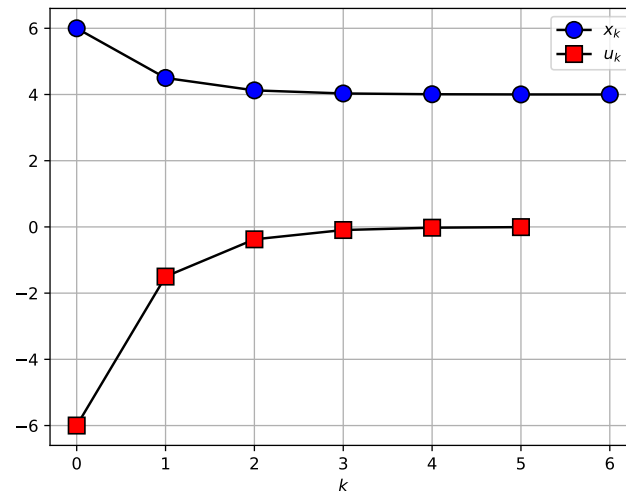
Figure 1: Plot of PID controller

Part I: Multiple-Choice

Question 1: ....

Suppose we want to apply the dynamical programming algorithm to chess [1]. This will lead to several practical problems, however, focusing just on the potential problems listed below, which one will be a main obstacle in terms of obtaining a near-perfect chess policy using the dynamical programming algorithm?

     A. Within a few iterations, the policy function μ_k will require too much memory to store

     B. Given a state $\boldsymbol{x}_k$, it is not practical to define the action spaces $\mathcal{A}_k(x_k)$

     C. It will require too much space to store the state space $\mathcal{S}_2$

     D. There is no reasonable choice of planning horizon N

     E. Don't know.

Question 2: ....

Consider PID control applied to steer a car along a straight track. The control signal u_k corresponds to the angle between the front wheel and the centerline of the track, the input signal x_k corresponds to the angle between the car body and the track in degrees, and the goal of the PID controller is to bring the angle between the car body and the track to a value of $x^* = 4$ degrees (corresponding to executing a turn). Figure 1 shows the behavior of both x_k and u_k at time steps $k = 0, 1, 2, \ldots$. Suppose the PID controller takes the form described in the lecture notes, and assume $K_d = K_i = 0$, which one of the following options are true?

     A. $K_p = 1$

     B. $K_p = 2$

     C. $K_p = 3$

     D. There is not enough information to determine the correct answer

     E. Don't know.

---

[1] We don't consider time control as part of the problem.

	Arm $a = 0$	Arm $a = 1$	Arm $a = 2$	Arm $a = 3$
$N_t(a)$	14	11	6	3
Total reward $S_t(a)$	45	31	15	5

Table 1: Outcome of simulating a k-armed bandit problem for a few iterations

Question 3: . . . .

Which one of the following options are correct about direct control using trapezoid collocation?

    A. It is an example of open-loop control

    B. It cannot take constraints into account

    C. It requires us to specify a fixed initial state $\boldsymbol{x}_0 = \boldsymbol{x}(t_0)$

    D. It can only be applied to quadratic cost functions

    E. Don't know.

Question 4: . . . .

Suppose iterative LQR is applied to steer the racecar from the car-example ([Her25, section 10.4.3]) through the track. Which of the following statements are true about the dimensions of the linearized matrices A_k and B_k?

    A. The dimension of A_k is 6×6 and the dimension of B_k is 6×2

    B. The dimension of A_k is 6×2 and the dimension of B_k is 6×1

    C. The dimension of A_k is 2×2 and the dimension of B_k is 2×6

    D. The dimension of A_k is 6×6 and the dimension of B_k is 2×2

    E. Don't know.

Question 5: . . . .

Consider a k-armed bandit problem with $k = 4$. Suppose we are at time step t, and we denote by $N_t(a)$ the total number of times we have pulled arm $a = 0, 1, 2, 3$ up to now, and by $S_t(a)$ the sum of all rewards we have obtained by pulling arm a, i.e. in the notation [SB18][Eq. 2.1]

$$S_t(a) = \{\text{Sum of rewards when } a \text{ taken prior to } t\} = \sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i = a}$$

$$N_t(a) = \{\text{Number of times } a \text{ taken prior to } t\} = \sum_{i=1}^{t-1} \mathbb{1}_{A_i = a}.$$

These values have been collected in table 1. Suppose we apply UCB1 to the problem using $c = 2$. Which arm will be selected next?

    A. UCB1 selects arm $A_t = 0$

    B. UCB1 selects arm $A_t = 1$

    C. UCB1 selects arm $A_t = 2$

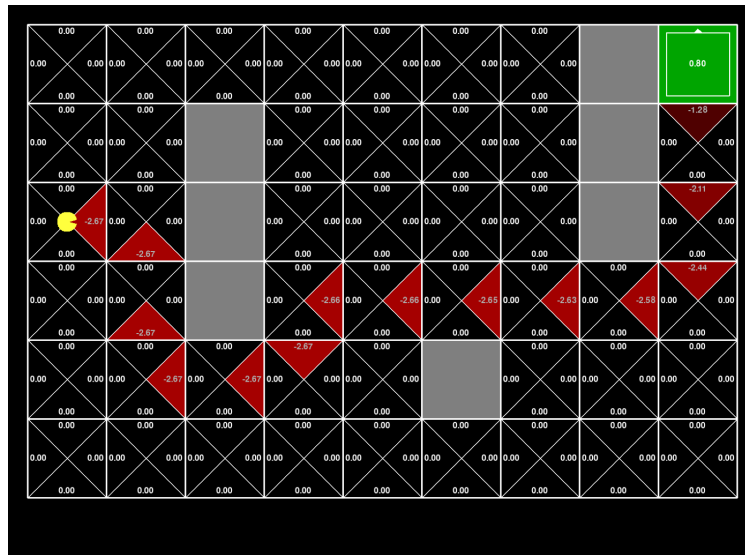   D. UCB1 selects arm $A_t = 3$

    E. Don't know.

Figure 2: A gridworld environment and the Q-values updated by applying first-visit Monte-Carlo for one episode. The plot is for illustration purposes, and the problem can be solved without consulting the particular values shown here.

Question 6: ....

Let v_*, q_* be the optimal value and action-value functions of an MDP, let π be any policy and finally let v_π and q_π be the value/action-value function associated with π. Which one of the following statements are true in general?

    A. $\max_s q_*(s, a) = v_*(a)$

    B. There is a policy π, a state s and an action a so that $q_*(s, a) < q_\pi(s, a)$

    C. For all π and a it is true that $q_*(s, a) > q_\pi(s, a)$

    D. There is a policy π and state s so that $\max_a q_*(s, a) = v_\pi(s)$

    E. Don't know.

Question 7: ....

We consider the familiar gridworld environment discussed in [Her25, section 4.2.4] shown in fig. 2. The agent receives a reward of $+1$ on completion (the upper-right square), and otherwise a living reward of -2. Recall that in first visit Monte-Carlo control (see [SB18, Section 5.3]), the action-values $Q(s, a)$ are computed as the average of the returns. Suppose that during a particular episode, the returns computed by first-visit monte carlo are $G_0, G_1, \ldots, G_{T-1}$ (for illustration purposes, fig. 2 shows a single update of first-visit monte carlo with our choice of rewards, and for particular choices of γ, α). What relationship holds true for the returns in the beginning of the episode, i.e. $t \leq T - 2$?

    A. $G_t = \gamma G_{t+1} - 2$

    B. $G_{t+1} = \alpha G_t - 2\gamma + G_T \gamma^T$

    C. $G_t = \alpha G_{t+1} \gamma^t$

    D. $G_t = -2\gamma^t + 1$

    E. Don't know.

Question 8: ....

Which of the following statements are true about TD(λ)?

    A. TD(λ) cannot be used with function approximators

    B. The role of the eligibility trace is to let reward obtained earlier in an episode affect the change in the value function later in the episode

    C. The eligibility trace cannot be negative

    D. The eligibility trace is a measure of the amount of reward obtained in a given state weighted by an exponential factor

    E. Don't know.

Question 9: ....

Which one of the following statements about Q-learning is correct?

    A. The first step in training a Q-learning agent is to compute the set of all states the agent can be in

    B. The Q-table $Q(s, a)$ in Q-learning is a measure of the reward the agent will obtain in the very next step multiplied by γ

    C. Q-learning still works if we initialize the Q-table to -1, i.e. $Q(s, a) = -1$ for all $s \in \mathcal{S}$

    D. When Q-learning is applied to a deterministic environment, the agent will follow a deterministic policy

    E. Don't know.

Part II: Conceptual questions

Question 10: ....
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:

$$\dot{x} = f(x, u) = 4ux \tag{1}$$

The first two questions will assume the problem has been discretized using a time constant of $\Delta = 0.5$ to yield states $x_0, x_1, x_2, \cdots$ and control signals $u_0, u_1, u_2, \dots$.

(a) Assume the problem is Euler discretized. Determine the discrete dynamics $\boxed{f_k}$ used to compute $x_{k+1} = f_k(x_k, u_k)$.

(b) Continuing the previous problem, suppose we wish to apply a LQR controller to control the system near a state $\bar{x}$. The system is therefore linearized around $\bar{x}$ and $\bar{u} = 1$ to give rise to the linearized dynamics $x_{k+1} = Ax_k + Bu_k + d$. Determine $\boxed{A}$, $\boxed{B}$ and $\boxed{d}$ in terms of $\bar{x}$.
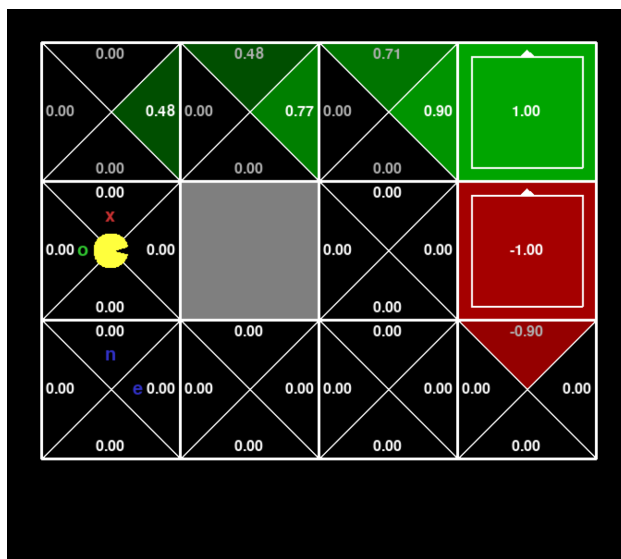
Figure 3: An example gridworld environment that Sarsa is applied to. The figure shows the current Q-values.

Question 11: ....

Suppose that Sarsa (using discount factor $\gamma = 1$, a learning rate $\alpha = 0.9$ and an exploration rate of $\varepsilon = 0.1$) is applied to the Gridworld MDP shown in fig. 3. The living reward is 0, and the dynamics is deterministic. Recall that pacman stay at the current state if he choose an action which moves him into a wall. The current state of the agent is as indicated in the figure.

(a) Suppose that in the next step of the Sarsa algorithm, the agent takes (and execute) the action North in the current position s. Upon taking this action, the Q-value associated with the red cross $Q(s, \text{North})$ will be updated by Sarsa.

   What are the possible value(s) of $\boxed{Q(s, \text{North})}$ after this step? (if there are more than one, list all of them).

(b) In the previous question, we took one step, performed a single action North, and updated one Q-value.

   In this question, suppose again that the agent starts in the position indicated in fig. 3 and assume Sarsa is applied to update the Q-values shown in the figure.

   Different sequences of future actions will result in different Q-values being updated. What are the $\boxed{\text{minimum number of steps}}$ which are required before the Q-value associated with the green circle can take a value different than 0, and $\boxed{\text{what actions}}$ will the agent take in this case? Give your answer as a list of actions.

(c) Sarsa learning has clearly not converged in the example shown above. However, assume we apply a more realistic version of Sarse where $\gamma = .95$, and where importantly α decrease to 0 at a rate satisfying the stochastic approximation conditions [SB18, Eq. (2.7)] for convergence so that the Q-values converge to their *true values* under Sarsa.

   Consider the two Q-value associated with moving North and East Q_n, Q_e, indicated by the blue n, e-letters in fig. 3. After convergence, it must be the case that either they will have the same value, $Q_n = Q_e$, or one will be greater than the other: $Q_n > Q_e$ or $Q_n < Q_e$. $\boxed{\text{State which is the case}}$ and provide a clear and specific argument for your answer.
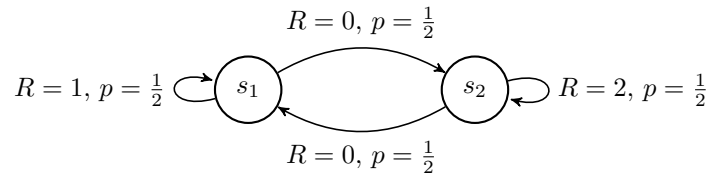
$$R = 0, p = \tfrac{1}{2}$$

$$R = 1, p = \tfrac{1}{2} \quad s_1 \qquad s_2 \quad R = 2, p = \tfrac{1}{2}$$

$$R = 0, p = \tfrac{1}{2}$$

Figure 4: A simple MRP

Question 12: ....

Consider a Markov Reward Process with two states s_1 and s_2 and a discount factor of $0 < \gamma < 1$ shown in fig. 4 [2]. With equal probability $\tfrac{1}{2}$, the system will either stay in the current state or transition to the other state. The MRP never terminates.

- When the system transition s_1 to s_2 (or s_2 to s_1) it will receive a reward of 0,
- If it transition from s_1 to s_1 it will receive a reward of 1,
- If it transition from s_2 to s_2 it will receive a reward of 2.

Since there are two states, the value function v_π takes two values $v_1, v_2 \in \mathbb{R}$ defined as: $v_1 = v_\pi(s_1)$ and $v_2 = v_\pi(s_2)$.

(a) Assume for a moment that $v_2 = \tfrac{5}{2}$ and $\gamma = \tfrac{2}{3}$. What is $\boxed{v_1}$? (i.e. the value function in s_1, $v_\pi(s_1)$)

(b) Ignore the previous question and consider the general form of the problem. As the name suggest, when Bellmans expectation equations are applied to this problem we obtain two equations. Write them as a linear system of the form $\boxed{\boldsymbol{b} = A\boldsymbol{v}}$ where $\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$. State what $\boxed{A}$ and $\boxed{\boldsymbol{b}}$ are as functions of γ.

---

[2]Recall a Markov Reward Process is a Markov decision process without actions, see [SB18, Example 6.2].

Part III: Programming questions

Question 13: ....

To solve this question, you should edit the file `irlc/exam/exam2023spring/question_inventory.py`. This problem focuses on a variant of the inventory control problem discussed in [Her25, section 5.1.2]. This inventory problem represents a flower-store such that x_k denotes the number of flower bouquets in stock at planning round k. The original inventory control model and the dynamical programming algorithm is included in the exam folder.

The following tasks can be solved by implementing suitable variants of the inventory control problem, and then applying dynamical programming to determine the optimal policy $\mu_0^*(x_0)$ and cost-function $J^*(x_0)$ in the starting state.

The flower store problem is equivalent to the inventory control problem on a horizon of N with two changes:[3]

- $g_k(x_k, u_k, w_k) = cu + |x_k + u_k - w_k|$
- The distribution of the number of items customers buy w_k is:

$$p_W(w_k = 0|x_k, u_k) = 0.1, \quad p_W(w_k = 1|x_k, u_k) = 0.3, \quad p_W(w_k = 2|x_k, u_k) = 0.6.$$

(a) Complete `def a_get_policy(N: int, c: float, x0 : int)`: This function is given a value of N, c and a starting state x_0, and should return the action the optimal policy computes in x_0, i.e. $\mu_0^*(x_0)$ as an `int`.

(b) Complete `def b_prob_one(N : int, x0 : int)`: For every policy and starting state x_0, there is a certain chance $p(x_N = 1|x_0)$ we will end up with a single item (bouquet) on the last day N when following the policy. The clerk operating the store would very much like to bring this last bouquet home with her, and so she is solely concerned with determining the policy which maximize $p(x_N = 1|x_0)$, i.e. the chance she can bring home a single bouquet at the end of the planning period.

Determine what this chance is when we follow the policy which is *solely* concerned with with maximizing the chance that $x_N = 1$. The function should accept N and x_0 as input argument, and return the value of $p(x_N = 1|x_0)$ as a `float`.

Hint: Alter the cost-functions so that the optimal solution maximize this probability. The Pacman-problems where we computed the probability of winning may provide inspiration.

---

[3]The expression $|x|$ return the absolute value, i.e. $|4| = |-4| = 4$

Question 14: ....

To solve this question, you should edit the file `irlc/exam/exam2023spring/question_lqr.py` . In this problem, we will consider the Pendulum-swingup task described in [Her25, section 10.4.1]. Recall that in this problem, the state of the pendulum is two-dimensional:

$$\boldsymbol{x} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

and under application of a control signal u, the state satisfies the differential equation:

$$\ddot{\theta} = \frac{g}{l}\sin(\theta) + \frac{u}{ml^2}, \quad g = 9.82,\ l = 1,\ m = 0.8. \tag{3}$$

(*The pendulum model is implemented as the class* `ContiniousPendulumModel` *in the file* `irlc/ex04/model_pendulum.py` *in the toolbox*). This model is assumed to be discretized using Euler discretization with a time constant of $\Delta = 0.5$ seconds, but *without* applying coordinate transformations. In other words, the discrete coordinates correspond to $\boldsymbol{x}_k = \begin{bmatrix} \theta_k \\ \dot{\theta}_k \end{bmatrix}$.

We want to eventually apply LQR to the problem. However, before we get to that we will consider how to simply apply discrete LQR to a comparable linear problem.

(a) Complete `def a_LQR_solve(a : float, x0 : np.ndarray)` : This function consider a discrete problem with cost matrices $Q = R = I$ and where the dynamics depends on a parameter a given by

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}\boldsymbol{x}_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u_k + \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The function should apply discrete LQR to the problem on a horizon sufficiently long to guarantee convergence[4] of the policy matrices L_0, $\boldsymbol{l}_0$ and return the first action u_0 the discrete LQR controller wish to execute in state $\boldsymbol{x}_0$. The action should be returned as a `float` .

(b) Complete `def b_linearize(theta : float)` : Return to the general formulation eq. (3). Given an angle θ, this function should linearize the discretized model around $\bar{\boldsymbol{x}} = \begin{bmatrix} \theta \\ 0 \end{bmatrix}$ and $\bar{u} = 0$, and return three numpy `ndarray` objects A, B and $\boldsymbol{d}$ of the right dimension corresponding to the linearized problem. I.e., it should hold approximately for states and actions close to the linarization point $\bar{\boldsymbol{x}}$, $\bar{u}$ that

$$\boldsymbol{x}_{k+1} \approx A\boldsymbol{x}_k + Bu_k + \boldsymbol{d}$$

(c) Complete `def c_get_optimal_linear_policy(x0 : np.ndarray)` : We are now ready to build our controller. Given input state $\boldsymbol{x}_0$, specified as an `np.ndarray` , this function should linearize the pendulum problem around $\bar{\boldsymbol{x}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\bar{u} = 0$, then solve the linearized problem using LQR (using a sufficiently long horizon to guarantee convergence and using $Q = R = I$), and then finally return the action the LQR controller will take in the $\boldsymbol{x}_0$ as a `float` .

---

[4]For instance $N = 100$

Question 15: ....

To solve this question, you should edit the file `irlc/exam/exam2023spring/question_bandit.py`. In this question, you will use bandit algorithms to determine which of k actions, corresponding to advertisements, are the best based on the clicks-per-hour they result in when presented to users. The bandit algorithm we will first consider will be the simple bandit algorithm presented in [SB18][Section 2.4]. The data will be presented to the algorithm in the shape of a list of actions $(a_1, a_2, \ldots, a_{t-1})$ (`actions`) and a list of corresponding rewards $(r_1, r_2, \ldots, r_{t-1})$ (`rewards`). The actions will be assumed to be integers $0, 1, \ldots, k-1$.

(a) Complete `def a_select_next_action_epsilon0(k : int, actions : list, rewards : list)`: Given a number of arms k, a list of $t-1$ actions `actions` and rewards `rewards`, this function should return the next action a_t as an `int` generated according to the bandit algorithm described in [SB18][Section 2.4] when trained on the data, but assuming we act greedily all the time: $\varepsilon = 0$.

(b) Complete `def b_select_next_action(k : int, actions : list, rewards : list, epsilon : float)`: Consider again the simple bandit algorithm in [SB18][Section 2.4]. In addition to the previous input, the function should now also accept a exploration rate $\varepsilon \geq 0$. The function should return the action a_t as an `int` generated according to the simple bandit algorithm.

(c) Complete `def c_nonstationary_Qs(k : int, actions : list, rewards : list, alpha : float)`: This function should implement the non-stationary simple bandit algorithm with α-soft updates described in [SB18][Section 2.5]. The inputs have the same meaning as before, but the function should also accept the learning rate α as an input argument. The function should return a dictionary which has actions a as keys, and their corresponding Q-value $Q(a)$ as values.

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, 15:47:08.