

**Written examination date:** March 4th, 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 1-4 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/midterm2023a/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/midterm2023a/midterm2023a_tests_grade.py` to generate your `.token`-file.

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

- The `irlc/exam/midterm2023a/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/midterm2023a/question_dp.py` source file containing your solution
- The `irlc/exam/midterm2023a/question_pid.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/midterm2023a` 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.

## Part I: Multiple-Choice

**Question 1:** .....

Which one of the following options are correct?

- A. A PID controller is an example of an open-loop controller
- B. The integral parameter  $K_i$  must always be positive  $K_i \geq 0$
- C. A PID controller is a model-based control method
- D. We can build a PID controller without specifying a cost-function**
- E. Don't know.

**Solution: The correct answer is D.**

- A. PID control is not open loop since it depends on the state
- B. The sign of the integral term (or any of the other terms) depends on the definition of  $u$
- C. The PID controller does not require a model.
- D. A PID controller does not need a model or a cost function.**

**Question 2:** .....

Which one of the following options are correct?

- A. Control problems where the continuous-time dynamics takes the form  $\ddot{x} = a\dot{x} + bx + c + u$  falls outside the scope of the linear quadratic regulator
- B. The linear-quadratic regulator is an example of model-free control
- C. In a linear-quadratic control problem of the form  $x_{k+1} = Ax_k + Bu_k$ , the matrices  $A$  and  $B$  must both be square.
- D. The cost-functions suitable for a linear-quadratic regulator can potentially produce negative values**
- E. Don't know.

**Solution: The correct answer is D.**

- A. The problem is linear if we use the state-representation  $\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$
- B. The linear-quadratic regulator explicitly plans using a model of the dynamics and cost (hence the name, linear-quadratic)
- C.  $A$  is square but  $B$  is only square when  $x_k$  and  $u_k$  have the same dimension.
- D. Since the cost-function allows a constant term it can easily be negative.**

**Question 3:** .....

Which one of the following options are correct?

- A. Euler-discretization is necessary if we want to apply LQR to a general continuous-time control problem.
- B. Euler-discretization is exact when both the dynamics and cost-function for the continuous-time control problem are linear
- C. To apply Exponential integration to a control problem of the form  $\dot{x} = Ax + Bu$ , it is necessary that  $A$  is invertible

Question 3 continues on the next page...

- D. Euler-discretization can be applied to time-dependent control problems
- E. Don't know.

**Solution: The correct answer is D.**

- A. Finite-horizon LQR can be applied to any problem with linear dynamics  $x_{k+1} = Ax_k + Bu_k$  without any reference to where these matrices come from (for instance, they can arise from exponential integration).
- B. This is not true since; c.f. the Harmonic Oscillator.
- C.  $A$  does not need to be invertible. In fact  $A$  is not invertible in the case  $\ddot{x}(t) = u(t)$  (see exercises to lecture 4).
- D. Euler-discretization is formulated for time-dependent control problems. Recall that the update is just  $x_{k+1} = x_k + \Delta f(x_k, u_k, t_k)$ .

**Question 4:** .....

Suppose the Dynamical Programming algorithm is applied to a problem where the following is known:

- $N = 10$
- The size of the action spaces are  $|\mathcal{A}_k(x_k)| = 4$
- The size of the states spaces are  $|\mathcal{S}_0| = 1$ ,  $|\mathcal{S}_N| = 2$  and otherwise  $|\mathcal{S}_k| = 10$
- There are exactly two random noise disturbances,  $w = 0$  and  $w = 1$ , available in any state/action combination:

$$P_W(w = 0|x, u) = P_W(w = 1|x_k, u_k) = \frac{1}{2}.$$

How many times does the dynamical programming algorithm need to evaluate  $f_k$  in order to find the optimal policy?

- A. 736
- B. 744
- C. 730
- D. 728**
- E. Don't know.

**Solution: The correct answer is D.**

At each time step  $k = 0, \dots, N - 1$ , the total number of evaluations of  $f_k$  is:

$$|\mathcal{S}_k| \times \text{actions} \times \text{disturbances}$$

The total is therefore  $(1 + (N - 1) \times 10)(4 \times 2) = 728$

## Part II: Conceptual questions

### Question 5: .....

Consider a control problem where a control signal  $u(t) \in \mathbb{R}$  is applied to a variable  $w(t) \in \mathbb{R}$ . The variable measures an angle, and it satisfies the following differential equation:

$$\ddot{w} = \cos(u + w) \quad (1)$$

We introduce a state  $\mathbf{x}(t) = \begin{bmatrix} w(t) \\ \dot{w}(t) \end{bmatrix}$  which allows us to re-write the system in the usual way as a first-order differential equation:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t)).$$

The problem is then discretized using Euler discretization with a time step of  $\Delta = 0.2$  to give states  $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots$ . Our goal is to bring the system to an angle  $w = \frac{\pi}{2}$  (where it should remain), corresponding to the goal state  $\mathbf{x}^* = \begin{bmatrix} \frac{\pi}{2} \\ 0 \end{bmatrix}$ .

- (a) If we succeed at bringing the system to the target state at time  $t'$ , how much control  $u(t')$  do we subsequently need to apply to keep the system at  $w(t) = \frac{\pi}{2}$ ? Provide an argument for your answer (Hint: Consider what it would mean for the system to stand still in terms of  $\mathbf{x}(t)$ )

**Solution:** At the goal state, the derivative of  $\mathbf{x}$  is  $\begin{bmatrix} 0 \\ \cos(\frac{\pi}{2} + u(t)) \end{bmatrix}$ . So assuming we don't apply any control the derivative is 0 and the system stays put. Therefore  $u(t') = 0$

- (b) Assume that the initial conditions at the starting time  $t_0 = 0$  is  $w(t = 0) = \dot{w}(t = 0) = 0$  and that no control signal is applied to the system ( $u(t) = 0, t \geq 0$ ). According to Euler discretization, what is the value of  $w$  at time  $t_k = \Delta$ ?

**Solution:** Euler discretization is:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \begin{bmatrix} x_1 \\ \cos(x_1 + u_k) \end{bmatrix}.$$

This means that for  $k = 0$  we get

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta \begin{bmatrix} x_1 \\ \cos(0) \end{bmatrix} = \begin{bmatrix} 0 \\ \Delta \end{bmatrix}$$

so that  $w_0 = 0$

- (c) Continuing the previous problem and still assuming no control signal is applied. According to Euler discretization, what is the value of  $w$  at time  $t_k = 2\Delta$ ?

**Solution:** Continuing the above, at  $k = 1$  we get:

$$\mathbf{x}_{k+1} = \begin{bmatrix} 0 \\ \Delta \end{bmatrix} + \Delta \begin{bmatrix} \Delta \\ \cos(0) \end{bmatrix} = \begin{bmatrix} \Delta^2 \\ 2\Delta \end{bmatrix}$$

I.e.  $w_1 = \Delta^2 = 0.04$ .

### Question 6: .....

Consider the dynamical programming setting where we plan over a horizon  $N > 0$ . We consider a problem where:

- The terminal cost function is

$$g_N(x_N) = x_N^2$$

Question 6 continues on the next page...

- For all  $k = 0, \dots, N - 1$  the dynamics is  $f_k(x, u, w) = ax + u - \lambda w$
- The noise disturbances are normally distributed with variance  $\sigma^2$ :

$$P_W(w|x, u) = \mathcal{N}(w \mid \mu = 0, \sigma^2 = 1)$$

- The states and actions are real numbers  $\mathcal{S}_k = \mathcal{A}_k(x_k) = \mathbb{R}$ .
- The non-terminal costs are only affected by  $u$ ,  $g_k(x, u, w) = u^2$ .

Thus, the relevant parameters of the problem are  $a$  and  $\lambda$ . We are concerned with optimal control.

- (a) Although the problem has been formulated as being about dynamical programming, note that the structure of the problem is that of a 1-dimensional LQR model. In the case where  $\lambda = 3$ , derive the expected future cost  $\mathbb{E}[g_N(x_N)|x_{N-1} = 0, u_{N-1} = 1]$  if we at time step  $k = N - 1$  are in state  $x_{N-1} = 0$  and take action  $u = 1$

**Solution:** The future cost is given by

$$\mathbb{E}[x_N^2] = \mathbb{E}_{\mathcal{N}(w|0,1)} [(ax + u - \lambda w)^2] = \mathbb{E} [(1 + \lambda w)^2] = 1 + \lambda^2 \mathbb{E}[w^2] + \mathbb{E}[2\lambda w] = 1 + \lambda^2$$

so the answer is  $\mathbb{E}[g_N(x_N)|x_{N-1} = 0, u_{N-1} = 1] = 10$

- (b) Derive an analytical expression for the optimal policy  $\mu_k(x_k)$  in time step  $k = N - 1$

**Solution:** Since it is known from the lecture notes that the optimal policy is not affected by noise we can set  $\lambda = 0$ . To proceed, we can either insert and optimize or run the LQR algorithm for 1 step. Using the LQR algorithm we get that:

$$V_N = Q_N = 2$$

$$R_k = 2 = R$$

$$A_k = a$$

$$B_k = 1$$

$$L_{N-1} = -(R_k + B_k^T V_{k+1} B_k)^{-1} (B_k^T V_{k+1} A_k) = -\frac{aQ_N}{R + Q_N} = -\frac{2a}{2 + 2} = -\frac{a}{2}$$

$$u_{N-1} = L_{N-1}x$$

Therefore  $\mu_{N-1}(x) = -\frac{a}{2}x$

## Part III: Programming questions

### Question 7: .....

To solve this question, you should edit the file `irlc/exam/midterm2023a/question_dp.py`. This problem exclusively focuses on the inventory control problem discussed in [Her25, section 5.1.2]. The inventory control model and the dynamical programming algorithm is included in the exam folder.

The dynamical programming algorithm determines the optimal cost-to-go function  $J_k^*$  for a dynamical programming problem as a list of dictionaries. We are interested in building a variant of the DP algorithm which computes the expected tail cost of a policy  $J_{\pi,k}(x_k)$ , also represented as a list of dictionaries.

- The problem itself will be represented as a `DPModel` instance
  - The policy  $\pi = (\mu_0, \mu_1, \dots, \mu_{N-1})$  is represented in the usual way as a list of length  $N - 1$ . Each element of this list corresponds to a  $\mu_k$  and is represented as a dictionary which maps states to actions
- (a) Complete `def a_expected_items_next_day(x : int, u : int)`: This function is given the starting state  $x_0$  and the first action  $u_0$ , and computes the expected value of the next state  $x_1$ :

$$\mathbb{E}[x_1 | x_0, u_0].$$

I.e., the function computes the expected amount of goods in the warehouse on day 1 given information about how much was in the warehouse on day 0 and how much we ordered  $u_0$ . *Hint: Recall that  $x_1 = f_0(x_0, u_0, w_0)$  where  $w_0$  is the random noise disturbance at time step  $k = 0$ .*

- (b) Complete `def b_evaluate_policy(pi : list, x0 : int)`: This function is given a policy  $\pi$  in the before-mentioned format and a starting state  $x_0$ , compute the expected tail cost  $J_{\pi,0}(x_0)$  when starting in state  $x_0$  at time  $k = 0$  and subsequently taking actions according to  $\pi$ . *Hint: You may find [Her25, section 6.3.1] to be of help.*

### Solution:

The solution can be found in this directory as a `.py` file.

**Question 8:** .....

To solve this question, you should edit the file `irlc/exam/midterm2023a/question_pid.py`. Your task is to implement a discrete PID control in a setting where you are given a **sequence** of states  $x_0, \dots, x_{N-1}$ , and have to compute the next action  $u_{N-1}$ . To avoid boundary issues, you can assume that  $N \geq 4$ . The discretization time is fixed at  $\Delta = 1$ .

The functions below all accept a sequence of states `xs`, represented as a list of numbers, and must compute the next action, represented as a `float`.

- (a) Complete `def a_pid_Kp(xs : list, xstar : float, Kp : float)`: Implement PID control to the sequence-of-states setting. The function is given a sequence of  $N$  states `xs` and a value of the proportionality term  $K_p$ , and should compute the next action. The parameter `xstar` corresponds to a target state  $x^*$ , but in this problem we assume that  $x^* = K_i = K_d = 0$ . The function should return a single real number corresponding to the next action  $u_{N-1}$ .
- (b) Complete `def b_pid_full(xs : list, xstar : float, Kp : float, Ki : float, Kd : float)`: This function is similar to the one you implemented in the previous question, but should also take the derivative term  $K_d$ , integral term  $K_i$  and target  $x^*$  into account.
- (c) Complete `def c_pid_stable(xs : list, xstar : float, Kp : float, Ki : float, Kd : float)`: This problem assumes that the states  $x_k$  are noisy and that we are particularly concerned about how this affects the derivative term in the PID controller.

Recall the contribution from the derivative term at time step  $k$  to the PID controller has the form:

$$K_p \cdot \{\dots\} + K_d e'_k + K_i \cdot \{\dots\}, \quad \text{where} \quad e'_k = \frac{e(t_k) - e(t_k - \Delta)}{\Delta}.$$

I.e., the term  $e'_k$  corresponds to the estimated derivative of the error. You should implement a variant of PID control where this term has been replaced with the average estimate over the last two time steps. The contribution from the  $K_d$  term should therefore be modified to be:

$$K_d \frac{e'_{k-1} + e'_k}{2}.$$

**Solution:**

The solution can be found in this directory as a `.py` file.



## References

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

*This line concludes the exam. Document build: 2025/04/30, 14:30:37.*