Лабораторна робота №3. Проєктування функції винагороди та політ до заданої точки

2 вересня 2026 р.

Мета

Навчитися будувати складену функцію винагороди для квадрокоптера й вимірювати внесок кожного доданка абляцією. Навчити політику польоту до заданої точки та розпізнавати погану винагороду за числами в логах.

Склад бригади і розподіл ролей

Робота виконується бригадою з трьох осіб. Кожен учасник відповідає за свою підсистему й захищає її окремо.

- **Учасник A — середовище і винагорода.** Реалізує середовище польоту до точки й складену винагороду, яка повертає і суму, і кожен доданок окремо; забезпечує вимикання будь-якого доданка нульовою вагою.
- **Учасник B — абляція.** Проводить шість абляційних експериментів, два з них із повним навчанням політики, і фіксує, що саме ламається в поведінці при вимиканні кожного доданка.
- **Учасник C — зіпсована винагорода.** Досліджує вагу штрафу за різкість керування й навмисно псує винагороду двома способами, показуючи, як політика експлуатує формулювання.

Інтеграційний крок виконують разом: усі запуски зводять в одну таблицю й формулюють, без яких доданків задача не розв'язується, а які змінюють лише манеру польоту.

Теоретичні відомості

Алгоритм максимізує суму винагород за епізод, і тільки її. Чого немає у винагороді, того для політики не існує, а записане з неправильною вагою виконується рівно настільки, наскільки вигідно. Винагорода і є постановкою задачі, тому помилку в ній не виправить довше навчання.

Стандартний набір доданків для польоту до точки має вигляд

$$r_t = -w_d \left\| p^* - p_t \right\| - w_\omega \left\| \omega_t \right\| - w_\theta \left(1 - R_{33}\right) - w_a \left\| a_t - a_{t-1} \right\| - w_c \mathbb{1}_{\text{crash}} + w_l \tag{1}$$

де p^*, p_t — цільова й поточна позиція, м; ω_t — кутова швидкість, рад/с; R_{33} — елемент матриці повороту (при рівному польоті $R_{33} = 1$, тому $(1 - R_{33})$ — відхилення від горизонту); a_t — дія: 4 числа CTBR (тяга і три кутові швидкості); $\mathbb{1}_{\text{crash}}$ — аварія: удар об підлогу чи стелю або нахил понад 1,2 рад. Базові ваги: $w_d = 1{,}00$, $w_\omega = 0{,}05$, $w_\theta = 0{,}50$, $w_a = 0{,}02$, $w_c = 10{,}0$, $w_l = 0{,}10$.

Головний внесок дає відстань до цілі, близько 74 % модуля суми. Решта доданків задає лише *манеру* польоту. Це формувальні доданки (**reward shaping**): у формі $F = \gamma\Phi(s_{t+1}) - \Phi(s_t)$ вони

не змінюють оптимуму. Штраф за аварію коштує близько 100 кроків польоту, тобто двох секунд. Без бонусу за життя всі доданки від'ємні, і найвигідніше впасти одразу.

Погану винагороду видають три ознаки:

- **локальний оптимум**: плато далеко від очікуваного, при $w_c = 100$ апарат висить під стелею;
- **уникання епізоду**: епізод коротшає, бо бонус за життя замалий;
- **експлуатація щілини** (**reward hacking**): сума росте, а задача не виконується. При $w_l = 1{,}0$ вигідніше простояти 400 кроків біля старту ($+400$), ніж летіти з 2 м (-150).

За самою сумою жодної з них не видно. Потрібен розклад на доданки.

Середовище

Python 3.12, `gym-pybullet-drones` (MIT) з усіма залежностями: `numpy`, `matplotlib`, `gymnasium`, `stable-baselines3`, `pybullet`. Усе встановлюється з PyPI і GitHub.

Одна тонкість встановлення: `pybullet` не має готових збірок під Python 3.12, тому компілюється з джерел і потребує компілятора C++. На Ubuntu це `sudo apt install build-essential`, на macOS — інструменти командного рядка Xcode. Якщо збірка не проходить, робіть роботу в хмарному ноутбуці з попередньо встановленим компілятором — умови завдання від цього не змінюються.

Порядок виконання

Обчислювальний бюджет. Відеокарта не потрібна: усі запуски розраховані на процесор звичайного ноутбука. Тримайтеся межі **30 хвилин на один запуск навчання**; кількість кроків добирайте під цю межу й фіксуйте її в ноутбуці однаковою для всіх порівнюваних запусків. Криву, що не встигла вийти на полицю, теж можна аналізувати — висновок робиться про *однаково* недонавчені політики.

1. **[разом]** Реалізувати середовище польоту до точки й складену винагороду за формулою (1) так, щоб функція повертала і суму, і кожен доданок окремо. Передбачити вимикання будь-якого доданка нульовою вагою.
2. **[A]** Написати опорну ПІД-політику й виконати опорний прогін із базовими вагами. Зберегти розклад винагороди за доданками як точку відліку.
3. **[B]** Провести абляцію: по черзі вимкнути кожен доданок і зафіксувати, що саме змінилося в поведінці та в числах. Два доданки на вибір дослідити з повним навчанням політики, решту — з опорною політикою.
4. **[C]** Дослідити вагу штрафу за різкість керування: обрати кілька значень і показати, як змінюються гладкість керування й час польоту.
5. **[C]** Зіпсувати винагороду щонайменше двома способами так, щоб політика почала експлуатувати формулювання. Для кожного випадку навести розклад за доданками й пояснити механізм.
6. **[разом]** Звести всі запуски в одну таблицю й сформулювати, без яких доданків задача не розв'язується, а які змінюють лише манеру польоту.

7. **[разом] Оформити ноутбук.** Зібрати роботу в один Jupyter-ноутбук: код кожного учасника у своїх комірках із короткими коментарями там, де це доречно. Обов'язково всередині: розклад винагороди за доданками, результати абляції, графіки для трьох значень ваги різкості й зведену таблицю запусків, а також висновки — три-п'ять тверджень із числами. Ноутбук має виконуватися від початку до кінця без ручних правок; у першій комірці вкажіть, хто з учасників за які комірки відповідав.

Контрольні запитання

1. Чому винагорода і є постановкою задачі?
2. Що станеться з політикою, якщо всі доданки від'ємні?
3. Чому відхилення від горизонту подають через R_{33}, а не через кут?
4. Скільком крокам польоту відповідає штраф за аварію в 10 одиниць?
5. Що таке reward shaping і коли він не змінює оптимуму?
6. Чим небезпечне надто складне формулювання винагороди?
7. Винагорода росте, але апарат не летить до цілі: що перевірите?
8. Чому при зростанні w_a з 0,02 до 0,20 страждає час, а не точність?
9. Навіщо абляція, якщо якість вимірюється сумою винагороди?
10. Політика висить на місці. Яку вагу зміните першою і в який бік?