02465: Introduction to reinforcement learning and control

Linear-quadratic problems in control

Tue Herlau

DTU Compute, Technical University of Denmark (DTU)

Lecture Schedule

Dynamical programming

1. The finite-horizon decision problem
   7 February
2. Dynamical Programming
   14 February
3. DP reformulations and introduction to Control
   21 February

Control

4. Discretization and PID control
   28 February
5. Direct methods and control by optimization
   7 March
6. **Linear-quadratic problems in control**
   14 March
7. Linearization and iterative LQR
   21 March

Reinforcement learning

8. Exploration and Bandits
   28 March
9. Bellmans equations and exact planning
   4 April
10. Monte-carlo methods and TD learning
    11 April
11. Model-Free Control with tabular and linear methods
    25 April
12. Eligibility traces
    2 May
13. Deep-Q learning
    9 May

Syllabus: `https://02465material.pages.compute.dtu.dk/02465public`
Help improve lecture by giving feedback on DTU learn

Reading material:

- [Her25, Chapter 16]

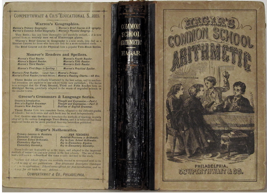
Learning Objectives

- Linear-quadratic regulator (LQR)
- Derivation of the LQR from DP
- Applications and variations

Practicals

- Project evaluations will be ready in about a week
- Part 2:
  - Less programming
  - A bit more emphasis on linear algebra; don't be afraid to write short answers if they are correct.
  - Be inspired by existing examples

Useful linear algebra



- A matrix A is **positive semi-definite** if it is symmetric and $\boldsymbol{x}^\top A \boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$
  - This means A behaves like a positive number: $ax^2 \geq 0$.
- if A is a symmetric matrix then:

$$\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} = \frac{1}{2}\left(\mathbf{x} + \mathbf{A}^{-1}\mathbf{b}\right)^T \mathbf{A}\left(\mathbf{x} + \mathbf{A}^{-1}\mathbf{b}\right) - \frac{1}{2}\mathbf{b}^T\mathbf{A}^{-1}\mathbf{b}$$

- This allows us to quickly find minimum

Recap: Dynamical programming algorithm

The Dynamical Programming algorithm

For every initial state x_0, the optimal cost $J^*(x_0)$ is equal to $J_0(x_0)$, and optimal policy π^* is $\pi^* = \{\mu_0, \dots, \mu_{N-1}\}$, computed by the following algorithm, which proceeds backward in time from $k = N$ to $k = 0$ and for each $x_k \in S_k$ computes

$$J_N(x_N) = g_N(x_N) \quad (1)$$

$$J_k(x_k) = \min_{u_k \in \mathcal{A}_k(x_k)} \mathbb{E}_{w_k} \{g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))\} \quad (2)$$

$$\mu_k(x_k) = u_k^* \quad (u_k^* \textit{ is the } u_k \textit{ which minimizes the above expression).} \quad (3)$$

Assumptions today

- For $k = 0, 1, \dots, N-1$

$$x_{k+1} = f_k(x_k, u_k, w_k) = A_k x_k + B_k u_k,$$
$$g_k(x_k, u_k, w_k) = \frac{1}{2} x_k^\top Q_k x_k + \frac{1}{2} u_k^\top R_k u_k,$$
$$g_N(x_k) = \frac{1}{2} x_N^\top Q_N x_N$$

- **Note: This is not the most general case, but will illustrate the main ideas**

Apply dynamical programming!

- Define $V_N \equiv Q_N$ and initialize:

$$J_N^*\left(\boldsymbol{x}_N\right) = \frac{1}{2}\boldsymbol{x}_N^T Q_N \boldsymbol{x}_N = \frac{1}{2}\boldsymbol{x}_N^T V_N \boldsymbol{x}_N$$

- DP iteration (start at $k = N - 1$)

$$J_k\left(\boldsymbol{x}_k\right) = \min_{\boldsymbol{u}_k} \mathbb{E}_{w_k}\left\{g_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k, w_k\right) + J_{k+1}\left(f_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k, w_k\right)\right)\right\}$$

- Remember to store optimal u_k^* as $\pi_k(x_k) = u_k^*$

LQR, simplified form

DP solution gives the controller:

1. $V_N = Q_N$
2. $L_k = -(R_k + B_k^T V_{k+1} B_k)^{-1} (B_k^T V_{k+1} A_k)$
3. $V_k = Q_k + L_k^T R_k L_k + (A_k + B_k L_k)^T V_{k+1} (A_k + B_k L_k)$
4. $\boldsymbol{u}_k^* = L_k \boldsymbol{x}_k$
5. $J_k^*(\boldsymbol{x}_k) = \frac{1}{2} \boldsymbol{x}_k^T V_k \boldsymbol{x}_k$

Double Integrator Example

- True dynamics

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u}(t) \tag{4}$$

- **Euler discretization** using $\Delta = 1$ System evolves according to:

$$\boldsymbol{x}_{k+1} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{=A} \boldsymbol{x}_k + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{=B} \boldsymbol{u}_k$$

- Cost function:

$$J(\boldsymbol{x}_0) = \sum_{k=0}^{N} \frac{1}{2\rho} x_{k,1}^2 + \sum_{k=0}^{N-1} \frac{1}{2} u_k^2$$

- Can be put into standard form using matrices/start position:

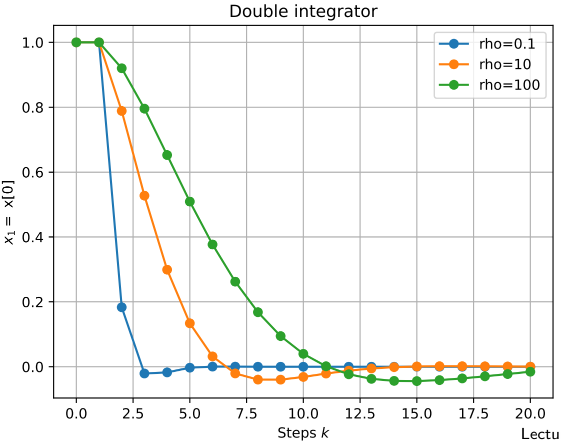
$$Q_k = Q_N = \begin{bmatrix} \frac{1}{\rho} & 0 \\ 0 & 0 \end{bmatrix} \quad R = 1$$

Exponential integrator

- Apply discrete LQR
- Simulate starting in $\boldsymbol{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ using policy

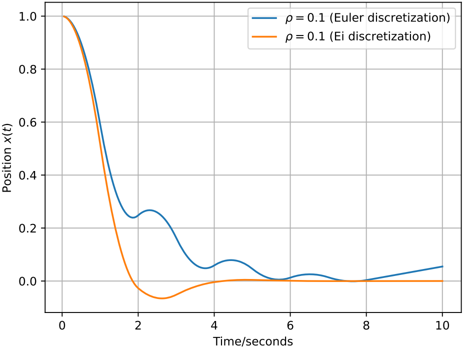
$$\pi_k(\boldsymbol{x}_k) = L_k \boldsymbol{x}_k$$

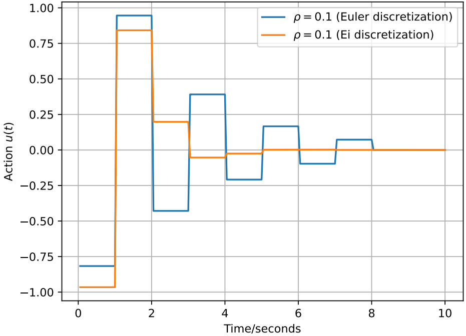
- What about the true system $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$?



Double integrator example

- **Blue:** LQR using Euler $\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u}_k$
- **Red:** LQR using Exponential $\boldsymbol{x}_{k+1} = e^{A\Delta}\boldsymbol{x}_k + A^{-1}\left(e^{A\Delta} - I\right) B\boldsymbol{u}_k$





- LQR is optimal in discrete problem
- Discrete controller can be bad in real problem (always check!)
- Always use EI for linear dynamics

Quiz: LQR

Consider a (generic) LQR problem of the form:

$$\boldsymbol{x}_{k+1} = A x_k + B u_k \tag{5}$$

$$\text{cost} = \sum_{k=0}^{N-1} \frac{1}{2} \boldsymbol{x}_k^\top Q \boldsymbol{x}_k + \frac{1}{2} R_0 \boldsymbol{u}_k^\top \boldsymbol{u}_k \tag{6}$$

Where $R_0 > 0$ is a constant. After LQR, the controller selects actions using $\boldsymbol{u}_k = L_k \boldsymbol{x}_k$. What do you think typically happens with the matrix L_k when $R_0 \to \infty$ (**very big** R_0)

a. The entries in L_k becomes very small, negative numbers

b. The entries in L_k becomes very big, positive numbers

c. It is not possible to say anything about the typical case

d. The entries in L_k gets closer to zero

e. Don't know.

Example: The locomotive

Steer locomotive (starting at $x = -1$) to goal ($x^* = 0$)

$$\ddot{x}(t) = \frac{1}{m} u(t) \tag{7}$$

Can be re-written as:

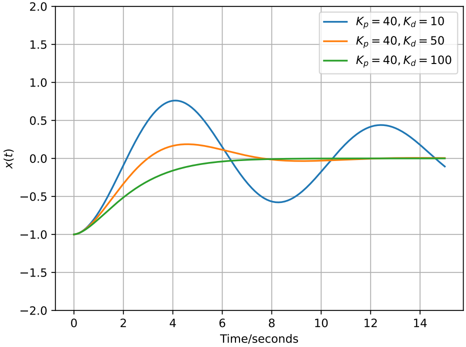
$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \tag{8}$$

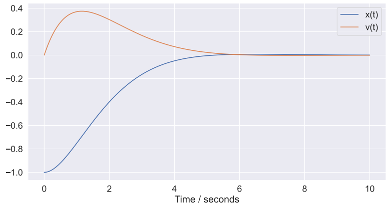
Discretized to $\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k$.

Locomotive: PID and LQR

$$e_k = x^* - x_k$$

$$u_k = e_k K_p + K_d \frac{e_k - e_{k-1}}{\Delta}$$





- Alternatively: Use a cost function $\sum_k \boldsymbol{x}_k^\top Q \boldsymbol{x}_k + \boldsymbol{u}_k^\top \boldsymbol{u}_k$ and use LQR!

`lecture_04_pid_d.py` `lecture_06_lqr_locomotive.py`

Planning on an infinite horizon

Recall LQR has the form:

1. $V_N = Q_N$
2. $L_k = -(R_k + B_k^T V_{k+1} B_k)^{-1}(B_k^T V_{k+1} A_k)$
3. $V_k = Q_k + L_k^T R_k L_k + (A_k + B_k L_k)^T V_{k+1}(A_k + B_k L_k)$
4. $\boldsymbol{u}_k^* = L_k \boldsymbol{x}_k$
5. $J_k^*(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T V_k \boldsymbol{x}_k$

- What happens if we repeat step 2 and 3 *many* times?
- The method will converge: $L_k \to L$
  - Select actions $\boldsymbol{u}_k = L\boldsymbol{x}_k$ (**"plan until convergence"**)
- If you think about it, this corresponds to planning on $N \to \infty$ horizon.
- **This is quite popular in control theory; what we will do in RL.**

Observations

- The cost term $\frac{1}{2}\boldsymbol{x}^\top Q\boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{R}\boldsymbol{u}$ is **smallest** when $\boldsymbol{x} = \boldsymbol{u} = \boldsymbol{0}$
- Implies that LQR will control system to state $\boldsymbol{x} = \boldsymbol{u} = \boldsymbol{0}$
- Suppose we want to drive system towards $\boldsymbol{x}_g, \boldsymbol{u}_g$?
  - Use $c(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_g)^T Q(\boldsymbol{x} - \boldsymbol{x}_g) + \frac{1}{2}(\boldsymbol{u} - \boldsymbol{u}_g)^T R(\boldsymbol{u} - \boldsymbol{u}_g)$
- more generally assume

$$c_k(\boldsymbol{x}_k, \boldsymbol{u}_k) = \frac{1}{2}\boldsymbol{x}_k^T Q_k \boldsymbol{x}_k + \frac{1}{2}\boldsymbol{u}_k^T R_k \boldsymbol{u}_k + \boldsymbol{u}_k^T H_k \boldsymbol{x}_k + \boldsymbol{q}_k^T \boldsymbol{x}_k + \boldsymbol{r}_k^T \boldsymbol{u}_k + q_k \quad (9)$$

$$c_N(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T Q_N \boldsymbol{x}_k + \boldsymbol{q}_N^T \boldsymbol{x}_k + q_N \quad (10)$$

and dynamics

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{d}_k$$

General discrete LQR algorithm

How to start living in luxury and never work again!

$$\cdots (V_{k+1} + \mu I) \cdots$$

1. $V_N = Q_N;\ \boldsymbol{v}_N = \boldsymbol{q}_N;\ v_N = q_N$

2.
$$L_k = -S_{\boldsymbol{uu},k}^{-1} S_{\boldsymbol{ux},k}$$
$$\boldsymbol{l}_k = -S_{\boldsymbol{uu},k}^{-1} S_{\boldsymbol{u},k}$$
$$S_{\boldsymbol{u},k} = \boldsymbol{r}_k + B_k^T \boldsymbol{v}_{k+1} + B_k^T V_{k+1} \boldsymbol{d}_k$$
$$S_{\boldsymbol{uu},k} = R_k + B_k^T V_{k+1} B_k$$
$$S_{\boldsymbol{ux},k} = H_k + B_k^T V_{k+1} A_k.$$

3.
$$V_k = Q_k + A_k^T V_{k+1} A_k - L_k^T S_{\boldsymbol{uu},k} L_k$$
$$\boldsymbol{v}_k = \boldsymbol{q}_k + A_k^T(\boldsymbol{v}_{k+1} + V_{k+1}\boldsymbol{d}_k) + S_{\boldsymbol{ux},k}^T \boldsymbol{l}_k$$
$$v_k = v_{k+1} + q_k + \boldsymbol{d}_k^T \boldsymbol{v}_{k+1} + \frac{1}{2}\boldsymbol{d}_k^T V_{k+1} \boldsymbol{d}_k + \frac{1}{2}\boldsymbol{l}_k^T S_{\boldsymbol{u},k}$$

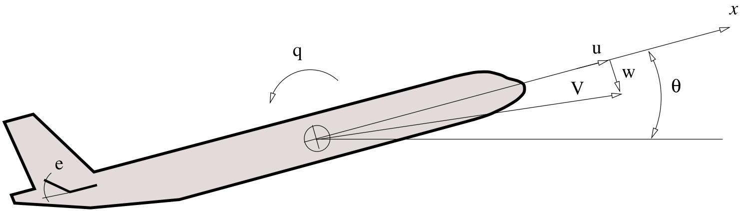
4. $\boldsymbol{u}_k^* = \boldsymbol{l}_k + L_k \boldsymbol{x}_k$

5. $J_k(\boldsymbol{x}_k) = \frac{1}{2}\boldsymbol{x}_k^T V_k \boldsymbol{x}_k + \boldsymbol{v}_k^T \boldsymbol{x}_k + v_k.$

Doctors hate this one weird trick!

$$V_k \leftarrow \frac{1}{2}(V_k^T + V_k)$$

Boing 747 Example



$$\underbrace{\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix}}_{} = \underbrace{\begin{bmatrix} -0.003 & 0.039 & 0. & -0.322 \\ -0.065 & -0.319 & 7.74 & 0. \\ 0.02 & -0.101 & -0.429 & 0. \\ 0. & 0. & 1. & 0. \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} u - u_w \\ w - w_w \\ q \\ \theta \end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix} 0.01 & 1. \\ -0.18 & -0.04 \\ -1.16 & 0.598 \\ 0. & 0. \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} e \\ t \end{bmatrix}}_{\boldsymbol{u}}$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \underbrace{\begin{bmatrix} 1. & 0. & 0. & 0. \\ 0. & -1. & 0. & 7.74 \end{bmatrix}}_{=P} \begin{bmatrix} u(t) - u_w(t) \\ w(t) - w_w(t) \\ q(t) \\ \theta(t) \end{bmatrix}$$

- y_1 and y_2 corresponds to the airspeed and climb rate.
- **Start**: $\boldsymbol{x} = 0$ (steady flight)
- **Want**: Air-speed of 10: $\boldsymbol{y}^* = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$

Approach

- Write dynamics as $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$
- Introduce cost function:

$$\int_0^{t_F} \left(\frac{1}{2}(\boldsymbol{y} - \boldsymbol{y}^*)^\top (\boldsymbol{y} - \boldsymbol{y}^*) + \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{u} \right) dt$$

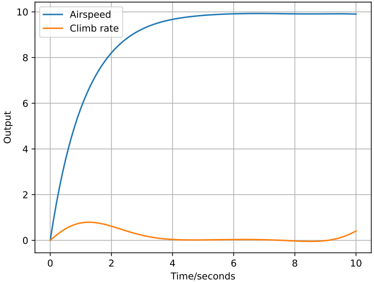
- Discretize dynamics using Exponential Integration to get $\boldsymbol{x}_{k+1} = \bar{A}\boldsymbol{x}_k + \bar{B}\boldsymbol{u}_k$
- Discretize cost to get one of the form

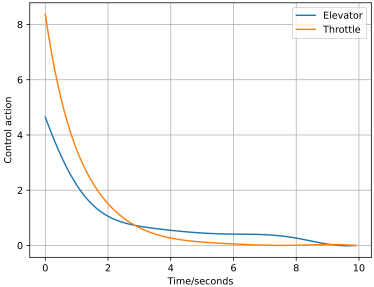
$$\sum_{k=0}^{\infty} \frac{1}{2}\boldsymbol{x}_k^\top Q \boldsymbol{x}_k + \boldsymbol{q}\boldsymbol{x}_k + q_0 + \frac{1}{2}\boldsymbol{u}_k^\top R \boldsymbol{u}_k$$

- Apply LQR!

Outcome and a Quiz

- Control law $\boldsymbol{u}_k = L x_k$





Left: airspeed and climb rate. **Right:** Elevator and throttle

Why does the output adjust quickly but fail to get entirely to the goal y^*?

a. Something bad happened to the dynamics with the exponential integration

b. The explanation has to do with planning on a finite horizon

c. The explanation is that R in $\boldsymbol{u}_k^\top R \boldsymbol{u}_k$ should be bigger

d. Don't know.

LQR with Additive Noise

- Consider the case where there is additive Gaussian noise:

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\omega}_k$$

- We can still solve the problem, and (amazingly!) the noise has **no influence** on the control law

$$\boldsymbol{u}_k = L_k \boldsymbol{x}_k$$

- LQR is robust to noise

Much more to LQR

- Stability/controllability of LQR?
    - **Important subject which we ignore**
- What if matrices A_k, B_k are random?
    - **This too can be solved[Ber05, Chapter 4]**
- What about partial observation?
    - **I.e. assume we observe $\boldsymbol{o}_k = D_k \boldsymbol{x}_k$[Ber05, Chapter 4]**
- What about constraints? What if we know $u_L \leq \boldsymbol{u}_k \leq u_B$?
- Euler integration is often not ideal.
    - **Alternatives including error analysis**

D.P. Bertsekas.
Dynamic Programming and Optimal Control.
Number v. 1 in Athena Scientific optimization and computation series.
Athena Scientific, 2005.

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