02465: Introduction to reinforcement learning and control

Discretization and PID control

Tue Herlau

DTU Compute, Technical University of Denmark (DTU)

Lecture Schedule

Dynamical programming

1. The finite-horizon decision problem
   2 February
2. Dynamical Programming
   9 February
3. DP reformulations and introduction to Control
   16 February

Control

4. **Discretization and PID control**
   23 February
5. Direct methods and control by optimization
   1 March
6. Linear-quadratic problems in control
   8 March
7. Linearization and iterative LQR
   15 March

Reinforcement learning

8. Exploration and Bandits
   22 March
9. Policy and value iteration
   5 April
10. Monte-carlo methods and TD learning
    12 April
11. Model-Free Control with tabular and linear methods
    19 April
12. Eligibility traces and value-function approximations
    26 April
13. Q-learning and deep-Q learning
    3 May

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

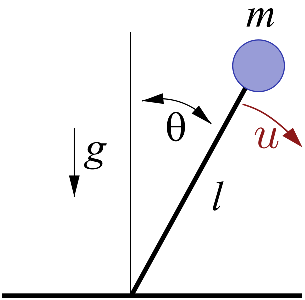
Reading material:

- [Her24, Chapter 12-14]

Learning Objectives

- Discretization of a control problem
- Control environments
- Exact solution for linear problems
- PID control

Example: The pendulum environment



If u is a torque applied to the axis of rotation θ then:

$$\ddot{\theta}(t) = \frac{g}{l}\sin(\theta(t)) + \frac{u(t)}{ml^2}$$

If $\boldsymbol{x} = \begin{bmatrix}\theta & \dot{\theta}\end{bmatrix}^T$ this can be written as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\theta} \\ \frac{g}{l}\sin(\theta) + \frac{u}{ml^2} \end{bmatrix} = f(\boldsymbol{x}, u) \quad (1)$$

`lecture_04_pendulum_random.py`

Dynamics

We assume the system we wish to control has dynamics of the form

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

- $\boldsymbol{x}(t) \in \mathbb{R}^n$ is a complete description of the system at t
- $\boldsymbol{u}(t) \in \mathbb{R}^d$ are the controls applied to the system at t
- The time t belongs to an interval $[t_0, t_F]$ of interest

Cost and policy

- The cost function will be of this form:

$$J_{\boldsymbol{u}}(\boldsymbol{x}, t_0, t_F) = \underbrace{c_F\left(t_0, t_F, \boldsymbol{x}\left(t_0\right), \boldsymbol{x}\left(t_F\right)\right)}_{\text{Mayer Term}} + \underbrace{\int_{t_0}^{t_F} c(\tau, \boldsymbol{x}(\tau), \boldsymbol{u}(\tau)) d\tau}_{\text{Lagrange Term}}$$

The continuous-time control problem

Given system dynamics for a system

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

Obtain $\boldsymbol{u} : [t_0; t_F] \to \mathbb{R}^m$ as solution to

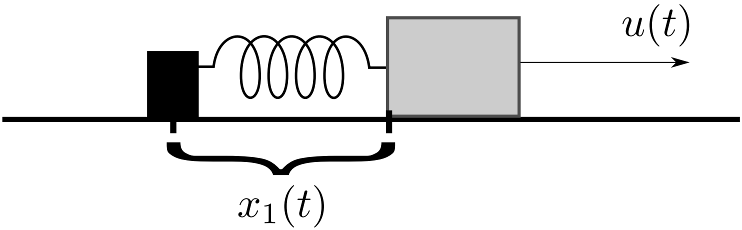
$$\boldsymbol{u}^*, \boldsymbol{x}^*, t_0^*, t_F^* = \arg\min_{\boldsymbol{x}, \boldsymbol{u}, t_0, t_F} J_{\boldsymbol{u}}(\boldsymbol{x}, \boldsymbol{u}, t_0, t_F).$$

(Minimization subject to all constraints)

Today:

- Linear-quadratic problems
- Discretization $t \to t_0, t_1, \dots, t_N$
- **Why?**
  - To build a `gymnasium` environment
  - To apply Dynamical Programming

Linear-quadratic problems: The harmonic oscillator



A mass attached to a spring which can move back-and-forth

$$\ddot{x}(t) = -\frac{k}{m}x(t) + \frac{1}{m}u(t) \tag{2}$$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \tag{3}$$

$$J = \int_0^{t_F} \left(\boldsymbol{x}(t)^\top \boldsymbol{x}(t) + u(t)^2 \right) dt. \tag{4}$$

`lecture_04_harmonic.py`

General linear-quadratic control

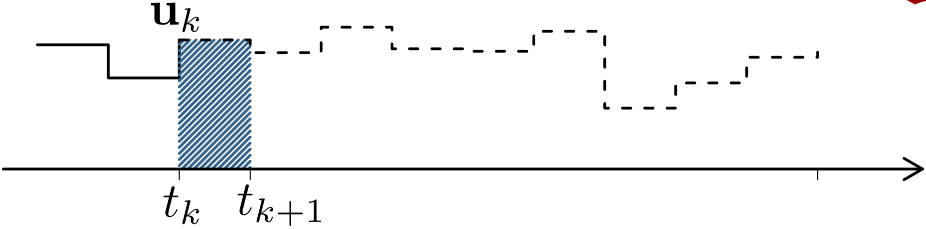
For $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times d}$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + \boldsymbol{d} \tag{5}$$

We assume $t_0 = 0$ and that the cost-function is quadratic:

$$J_{\boldsymbol{u}}(\boldsymbol{x}_0, t_F) = \frac{1}{2} \int_0^{t_f} \boldsymbol{x}^T(t) Q \boldsymbol{x}(t) + \boldsymbol{u}^T(t) R \boldsymbol{u}(t) dt \tag{6}$$

Discretization



- Euler-integration will be used to discretize the model:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ &= \boldsymbol{x}_k + \Delta \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) \end{aligned}$$

$$J_{\boldsymbol{u}=(\boldsymbol{u}_0, \boldsymbol{u}_1, \dots, \boldsymbol{u}_{N-1})}(\boldsymbol{x}_0) = c_f(t_0, \boldsymbol{x}_0, t_F, \boldsymbol{x}_F) + \sum_{k=0}^{N-1} c_k(\boldsymbol{x}_k, \boldsymbol{u}_k)$$

$$c_k(\boldsymbol{x}_k, \boldsymbol{u}_k) = \Delta c(\boldsymbol{x}_k, \boldsymbol{u}_k).$$

- The discrete model is deterministic but approximate:
  Open-loop no longer optimal

Variable transformation

- It is common to consider variable transformations. For the pendulum:

$$\phi_x : \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \mapsto \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \dot{\theta} \end{bmatrix}. \tag{7}$$

(avoids periodiodic)

- For control signal $-U < u < U$:

$$\phi_u : \begin{bmatrix} u \end{bmatrix} \mapsto \begin{bmatrix} \tanh^{-1} \frac{u}{U} \end{bmatrix}. \tag{8}$$

(No longer constrained)

- The update equations in the discrete coordinates $\boldsymbol{x}_k$, $\boldsymbol{u}_k$ are:

$$\boldsymbol{x}_{k+1} = \phi_x \left(\phi_x^{-1}(\boldsymbol{x}_k) + \Delta \boldsymbol{f}(\phi_x^{-1}(\boldsymbol{x}_k), \phi_u^{-1}(\boldsymbol{u}_k), t_k) \right) \tag{9}$$

$$= \boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{10}$$

Quiz: Discretization

Consider the pendulum: If $\boldsymbol{x} = \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^T$ this can be written as

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\theta} \\ \frac{g}{l} \sin(\theta) + \frac{u}{ml^2} \end{bmatrix} = f(\boldsymbol{x}, u)$$

What is the Euler discretization update using the convention $\boldsymbol{x}_k = \begin{bmatrix} \theta_k \\ \dot{\theta}_k \end{bmatrix}$?

a. $\begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix} = \Delta \begin{bmatrix} \theta_k + \dot{\theta}_k \\ \frac{g}{l} \sin \theta_k + \frac{u_k}{ml^2} \end{bmatrix}$

b. $\begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} \theta_k + \Delta \dot{\theta}_k \\ \dot{\theta}_k + \Delta \left(\frac{g}{l} \sin \theta_k + \frac{u_k}{ml^2} \right) \end{bmatrix}$

c. $\begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix} = \Delta \begin{bmatrix} \theta_k \\ \frac{g}{l} \sin \theta_{k+1} + \frac{u_k}{ml^2} \end{bmatrix}$

d. $\begin{bmatrix} \theta_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} \Delta \theta_{k+1} + \dot{\theta}_k \\ \Delta \dot{\theta}_{k+1} + \frac{g}{l} \sin \theta_k + \frac{u_k}{ml^2} \end{bmatrix}$

Exponential Integration of linear models

Recall that general linear dynamics has the form

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) + \boldsymbol{d} \tag{11}$$

Euler integration would suggest:

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \Delta f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ &= (I + \Delta A)\boldsymbol{x}_k + \Delta B\boldsymbol{u}_k + \Delta\boldsymbol{d}\end{aligned}$$

In fact, the following is an **exact** solution (see [Her24, section 12.1])

$$\boldsymbol{x}_{k+1} = e^{A\Delta}\boldsymbol{x}_k + A^{-1}(e^{A\Delta} - I)B\boldsymbol{u}_k + A^{-1}(e^{A\Delta} - I)\boldsymbol{d} \tag{12}$$

(The symbol $e^A \approx I + A + \frac{1}{2}A^2 + \cdots$ is the matrix exponential)

Implementation

- You still only implement a `ControlModel` class (as last week)
- Creating a discrete model and an environment is automatic
- See the online documentation for week 4.

Approaches to control

- Rule-based methods (build $\boldsymbol{u}(t) = \pi(\boldsymbol{x}, t)$ directly)
- Optimization-based methods:

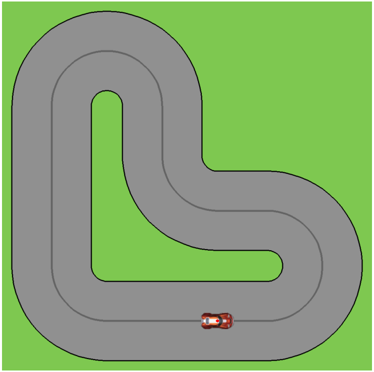
$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} J_{\boldsymbol{u}}(\boldsymbol{x}_0)$$

- DP-inspired planning methods

PID Control

Consider a water-heater where we apply heat u to keep temperature x at a desired level x^*

- If $x < x^*$ apply more u
- If $x > x^*$ apply less u



- If left-of-centerline turn wheel u right
- If right-of-centerline turn wheel u left

`lecture_01_car_random.py` `lecture_04_car_basic_pid.py`

Example: The locomotive

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

$$\ddot{x}(t) = \frac{1}{m} u(t) \tag{13}$$

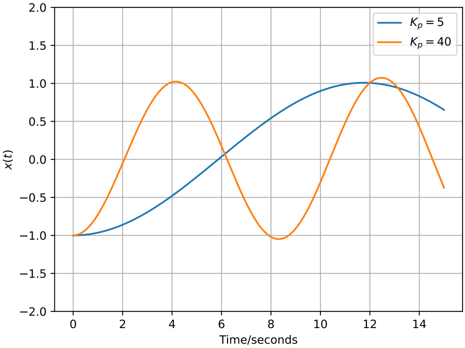
Or alternatively:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \tag{14}$$

P is for proportionality

Idea: If $x < x^*$, increase u proportional to $x^* - x$:

$$e_k = x^* - x_k$$
$$u_k = e_k K_p$$



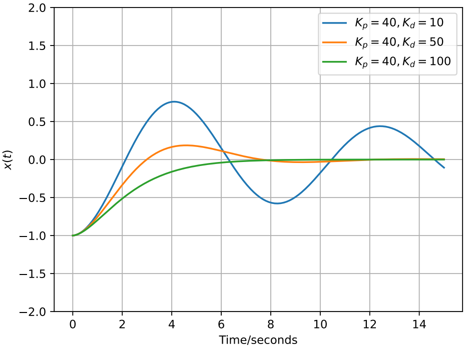
`lecture_04_pid_p.py`

D is for derivative

Idea: Slow down approach when e changes

$$e_k = x^* - x_k$$

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



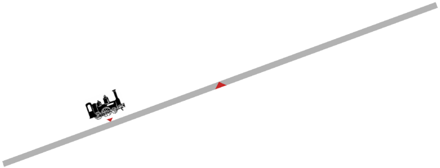
`lecture_04_pid_d.py`

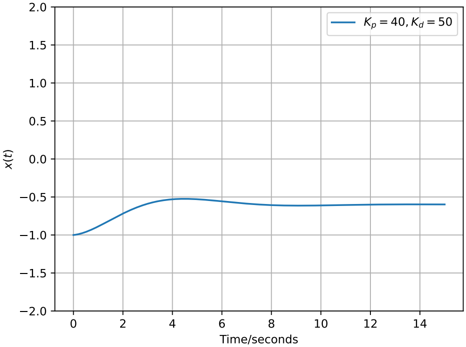
Droop

Using same controller as before on an inclined plane

$$e_k = x^* - x_k$$

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



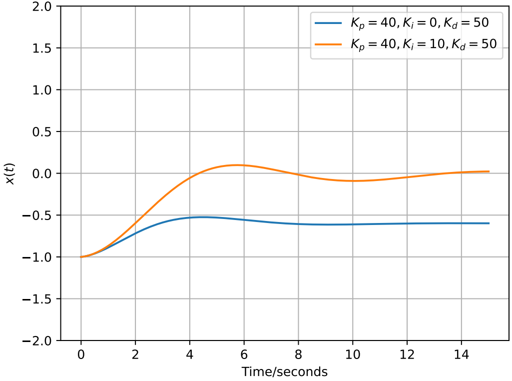


`lecture_04_pid_iA.py`

I in PID fixes droop

We fix droop by accumulating the total drop and adding it to u:

$$e_k = x^* - x_k$$
$$I_k = I_{k-1} + \Delta e_k$$
$$u_k = e_k K_p + K_d \frac{e_k - e_{k-1}}{\Delta} + I_k$$



`lecture_04_pid_iB.py` **(Should we limit the maximum value of I_k?)**

PID controller

Algorithm 1 PID controller

```
K_p > 0 and K_i, K_d ≥ 0
Δ time between observations x_k (discretization)
x* Control target
e^prev ← 0                                   ▷ Previous value of error
function POLICY(x_k)                         ▷ PID Controller called with observation x_k
    e ← x* − x_k                             ▷ Compute error
    I ← I + Δe                               ▷ Update integral term
    u ← K_p e + K_i I + K_d (e − e^prev)/Δ   ▷ PID control signal
    e^prev ← e                               ▷ Save current error for next iteration
    return u
end function
```

Quiz: PID control

Suppose the pendulum is discretized using a time discretization constant of $\Delta = \frac{1}{2}$ seconds. If the angle is $w_k = 2$ and a PID controller is applied with $K_p = 2$ and $K_d = K_i = 0$ (and a target of 0 degrees), what is the control output?

a. $u_k = -4$

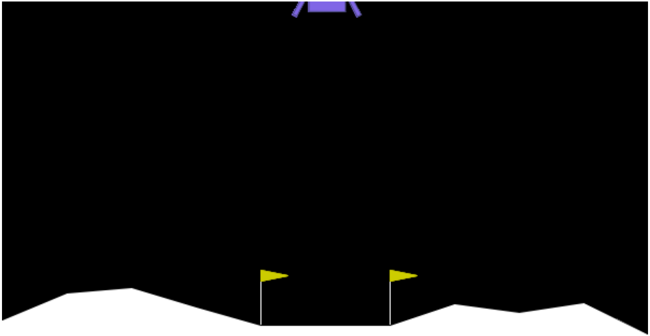
b. $u_k = 8$

c. $u_k = 4$

d. $u_k = -8$

e. Don't know.

Example:



lecture_04_cartpole_A.py , lecture_04_lunar.py

Tue Herlau.
Sequential decision making.
(Freely available online), 2024.