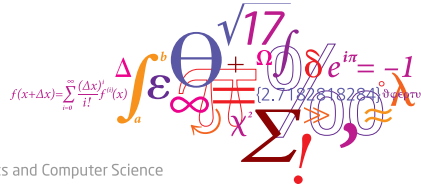


02465: Introduction to reinforcement learning and control

Discretization and PID control

Tue Herlau

DTU Compute, Technical University of Denmark (DTU)



DTU Compute
Department of Applied Mathematics and Computer Science

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:

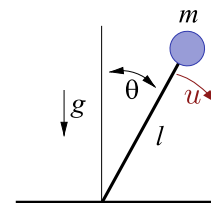
- [Her25, Chapter 12-14]

Learning Objectives

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

The control problem

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 $\mathbf{x} = [\theta \quad \dot{\theta}]^T$ this can be written as

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

🔗 [lecture_04_pendulum_random.py](#)

The control problem

Dynamics

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

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

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

The control problem

Cost and policy

- The cost function will be of this form:

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

The continuous-time control problem

Given system dynamics for a system

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

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

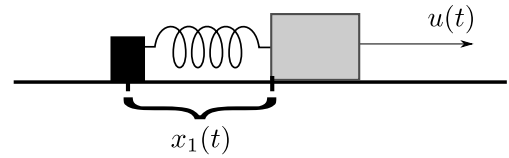
$$\mathbf{u}^*, \mathbf{x}^*, t_0^*, t_F^* = \arg \min_{\mathbf{x}, \mathbf{u}, t_0, t_F} J_{\mathbf{u}}(\mathbf{x}, \mathbf{u}, t_0, t_F).$$

(Minimization subject to all constraints)

Today:

- Linear-quadratic problems
- Discretization $t \rightarrow t_0, t_1, \dots, t_N$
- **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

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

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \quad (3)$$

$$J = \int_0^{t_F} (\mathbf{x}(t)^\top \mathbf{x}(t) + u(t)^2) dt. \quad (4)$$

lecture_04_harmonic.py

General linear-quadratic control

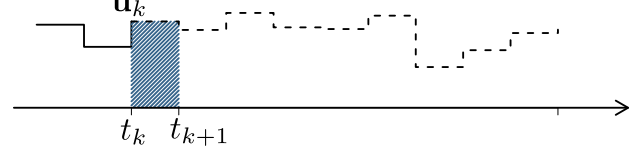
For $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times d}$

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

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

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

Discretization



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

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

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

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

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

Quiz: Discretization

Consider the pendulum: If $\mathbf{x} = [\theta \ \dot{\theta}]^T$ this can be written as

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

What is the Euler discretization update using the convention $\mathbf{x}_k = [\theta_k \ \dot{\theta}_k]^T$?

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

Variable transformation

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

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

(avoids periodicity)

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

$$\phi_u : [u] \mapsto [\tanh^{-1} \frac{u}{U}]. \quad (8)$$

(No longer constrained)

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

$$\mathbf{x}_{k+1} = \phi_x(\phi_x^{-1}(\mathbf{x}_k) + \Delta \mathbf{f}(\phi_x^{-1}(\mathbf{x}_k), \phi_u^{-1}(\mathbf{u}_k), t_k)) \quad (9)$$

$$= \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k) \quad (10)$$

Exponential Integration of linear models

Recall that general linear dynamics has the form

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

Euler integration would suggest:

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

In fact, the following is an **exact** solution (see [Her25, section 12.1])

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

(The symbol $e^A \approx I + A + \frac{1}{2}A^2 + \dots$ 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 $\mathbf{u}(t) = \pi(\mathbf{x}, t)$ directly)
- Optimization-based methods:

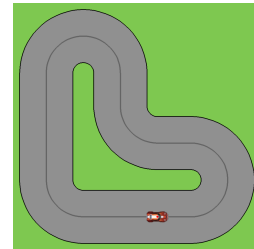
$$\mathbf{u}^* = \arg \min_{\mathbf{u}} J_{\mathbf{u}}(\mathbf{x}_0)$$

- 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 ($x^* = 0$)

$$\dot{x}(t) = \frac{1}{m}u(t) \quad (13)$$

Or alternatively:

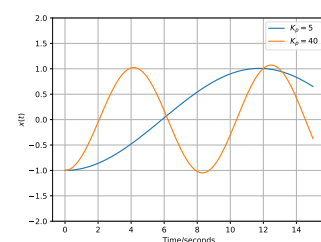
$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \quad (14)$$

P is for proportionality

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

$$e_k = x^* - x_k$$

$$u_k = e_k K_p$$



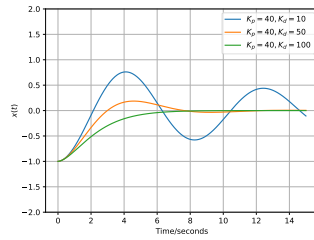
🔗 `lecture_04_pid_p.py`

D is for derivative

Idea: Slow down approach when e changes

$$e_k = x^* - x_k$$

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



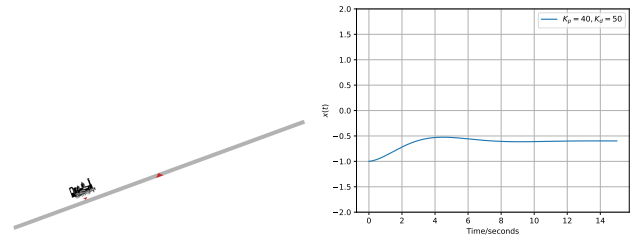
lecture_04_pid_d.py

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

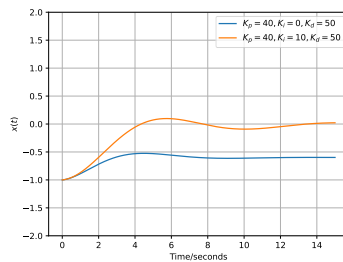
I in PID fixes droop

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 ?)

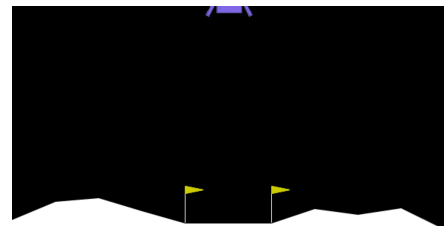
PID controller**Algorithm 1** PID controller

- 1: $K_p > 0$ and $K_i, K_d \geq 0$
- 2: Δ time between observations x_k (discretization)
- 3: x^* Control target
- 4: $e^{\text{prev}} \leftarrow 0$ ▷ Previous value of error
- 5: **function** POLICY(x_k) ▷ PID Controller called with observation x_k
- 6: $e \leftarrow x^* - x_k$ ▷ Compute error
- 7: $I \leftarrow I + \Delta e$ ▷ Update integral term
- 8: $u \leftarrow K_p e + K_i I + K_d \frac{e - e^{\text{prev}}}{\Delta}$ ▷ PID control signal
- 9: $e^{\text{prev}} \leftarrow e$ ▷ Save current error for next iteration
- 10: **return** u
- 11: **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), 2025.