2A-SG8-MPC

Model Predictive Control

CentraleSupélec

Romain.Bourdais@centralesupelec.fr

CentraleSupélec

CHAPTER 3

CONNECTION WITH CONTROL THEORY – LINK WITH LQR

2A-SG8-MPC
WOOCLAP WILL BE WITH US

3.1 MOTIVATION EXAMPLE

STABILITY ISSUES? IMPACT OF N_p AND λ

Exercice 3.1

Let us consider the following discrete-time linear system

$$\left\{x(k+1) = \begin{pmatrix} 1.1 & 2 \\ 0 & 0.95 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 0.0787 \end{pmatrix} u(k)\right.$$

and the criterion to optimize is

$$J^{N_p} = \sum_{i=1}^{N_p} \left(x^\intercal(k+i|k) \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} x(k+i|k) + \lambda u^2(k+i-1|k) \right)$$

1. Try several configurations for different values of N_p and λ and compare with the Infinite horizon solution

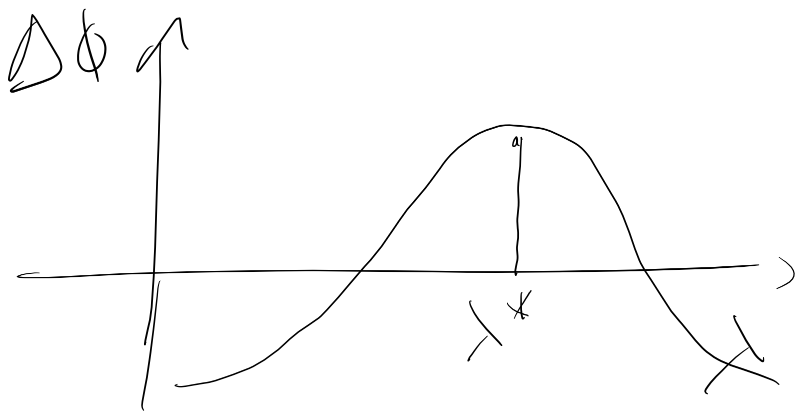
IMPACT OF N_p AND λ
IMPACT ON THE STABILITY MARGINS

Impact of N_p on stability:

$N_p \nearrow \implies$ stability increases

$\implies$ computational increases.

trade-off.



3.2 ANOTHER WAY TO SOLVE THE PROBLEM: DYNAMIC PROGRAMMING APPROACH

A WORD ON DYNAMIC PROGRAMMING PRINCIPLE

- Problem to solve depends on the data: if the solution exists, it should be a function of these data

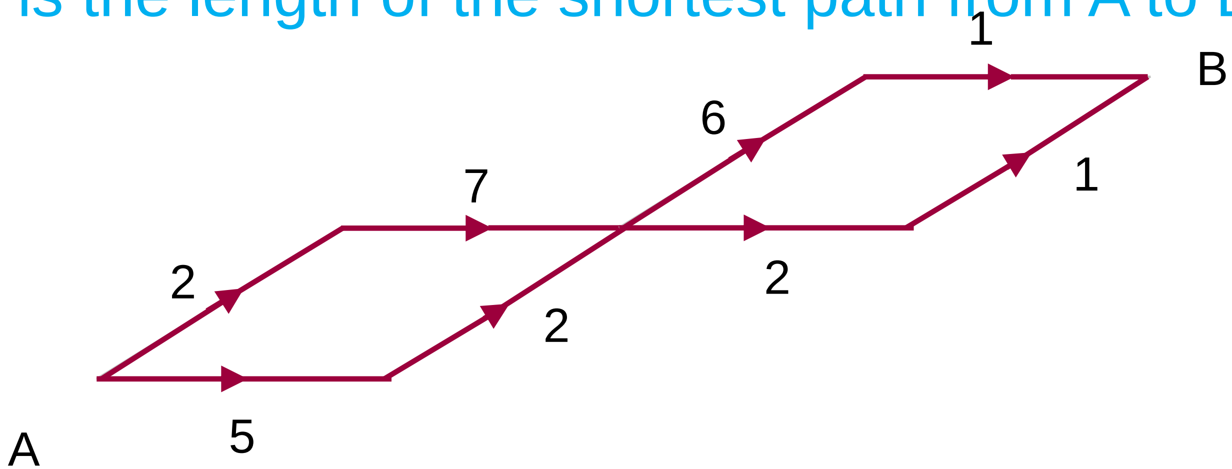
- The objective is then to have an explicit relation between the solution and the data

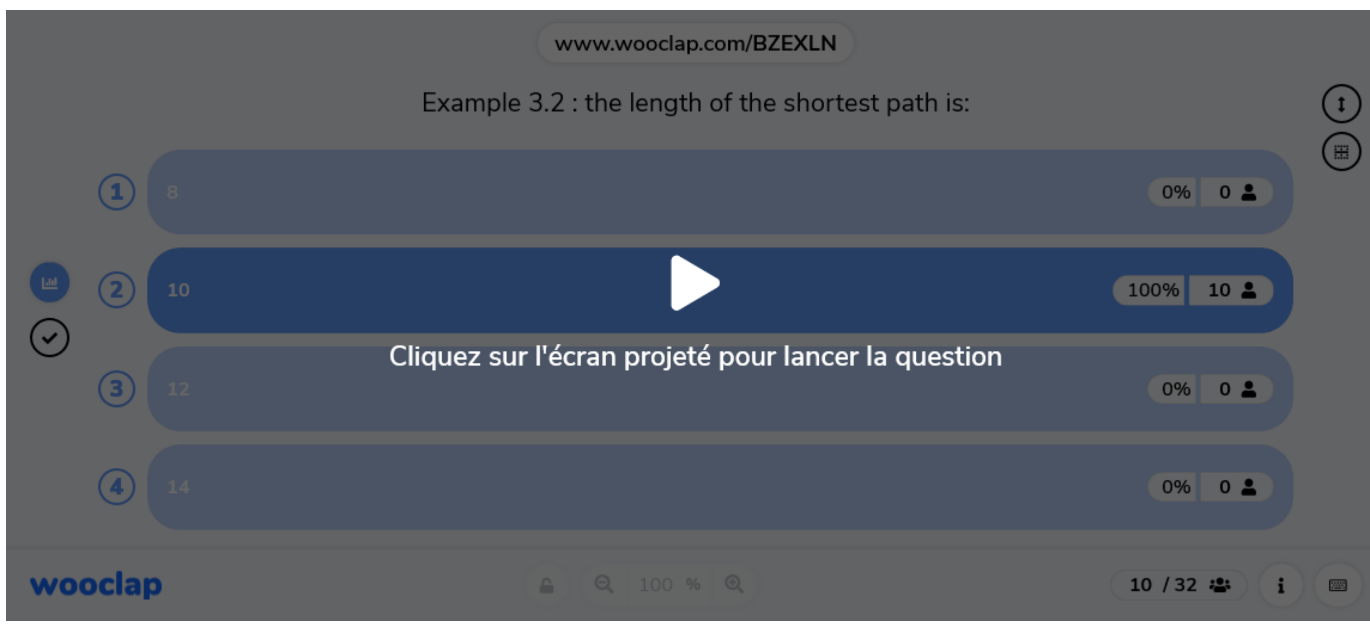
⇒ The problem will be solved for all the possible data

⇒ The solution will be extracted from the exact data of the problem

A WORD ON DYNAMIC PROGRAMMING
EXAMPLE 3.2

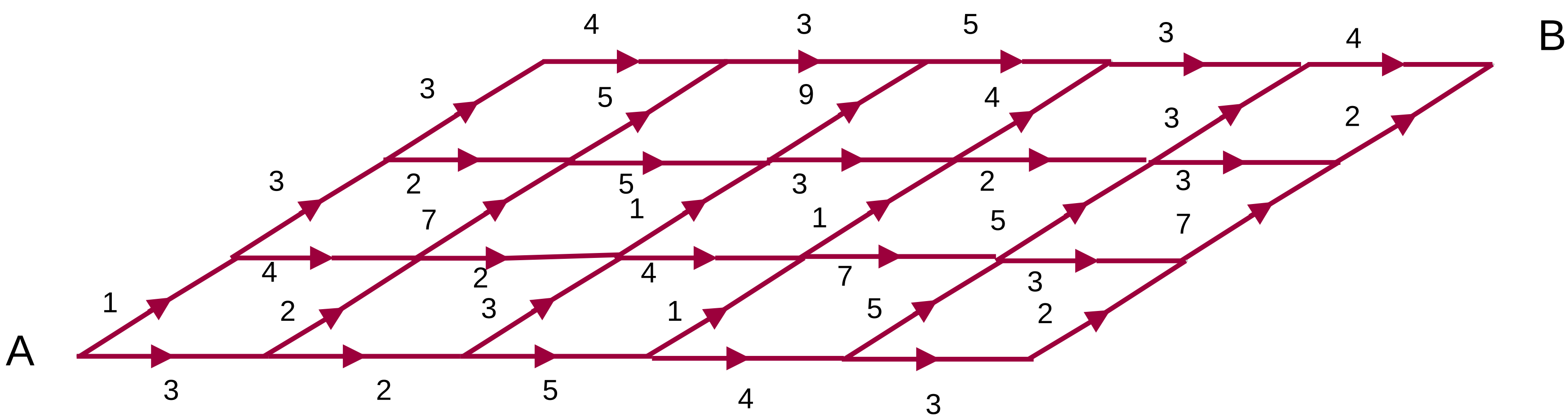
What is the length of the shortest path from A to B ?

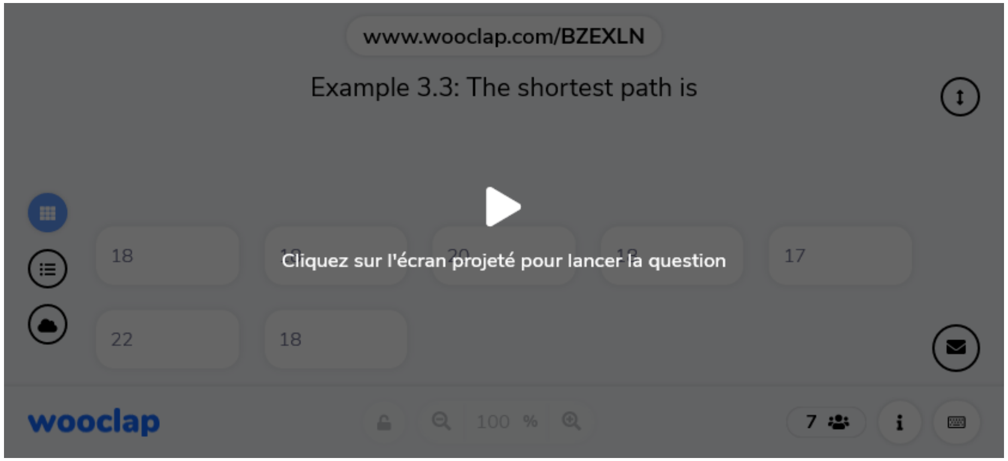




A WORD ON DYNAMIC PROGRAMMING
EXAMPLE 3.3

What is the length of the shortest path from A to B ?





We need a good methodolgy !

A WORD ON DYNAMIC PROGRAMMING
THE MAIN PRINCIPLE: BELLMAN

动态规划原则

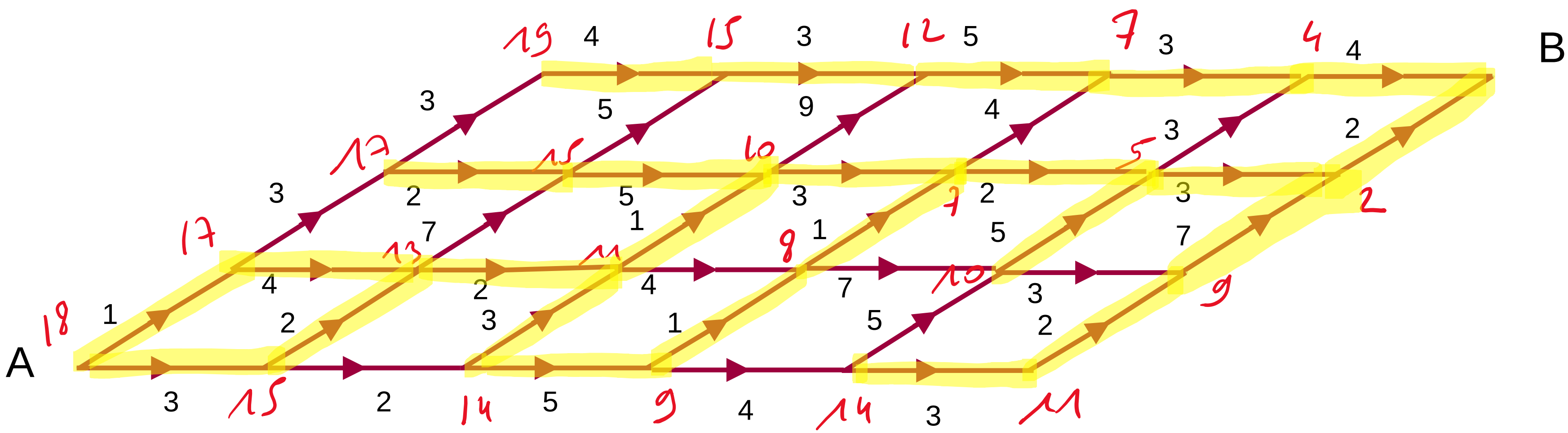
- $\hat{R}(i)$: Minimum cost from i to B
- r_{ij} : the cost to go from i to j

$$\hat{R}(i) = \min_{j\ admissible} \{r_{ij} + \hat{R}(j)\}$$

We will go backward, starting with $\hat{R}(B) = 0$

A WORD ON DYNAMIC PROGRAMMING
EXAMPLE 3.3

What is the length of the shortest path from A to B ?



And the winner is:

A WORD ON DYNAMIC PROGRAMMING
APPLICATION TO CONTROL PROBLEM

$$x(k+1) = f\big(x(k), u(k)\big)$$

$$\boxed{J^{N_p} = \min_{U} \sum_{i=0}^{N_p} l(x(k+i), u(k+i)) \;\; = \tilde{J}(x(k), 0)}$$

$$\begin{aligned}
\tilde{J}(x(k+j), j) \quad &= \quad \min_{\{u(k+i)\}} \sum_{i=j}^{N_p} l(x(k+i), u(k+i)) \\
&= \quad \min_{u(k+j)} \left\{ l\big(x(k+j), u(k+j)\big) + \sum_{i=j+1}^{N_p} l\big(x(k+i), u(k+i)\big) \right\} \\
&= \quad \min_{u(k+j)} \{ l(x(k+j), u(k+j)) + \tilde{J}(f(x(k+j), u(k+j)), j+1) \}
\end{aligned}$$

Initialization : $\tilde{J}\big(x(k+N_p), N_p\big) = 0$

APPLICATION: LTI, QUADRATIC COST

$$x(k+1) = Ax(k) + Bu(k)$$

$$J^{N_p} = \underbrace{\frac{1}{2}\sum_{i=0}^{N_p-1}\left(x(k+i)^\top Q x(k+i) + u(k+i)^\top R u(k+i)\right)}_{\text{running cost}} + \underbrace{\frac{1}{2}x(k+N_p)^\top P_{N_p} x(k+N_p)}_{\text{terminal cost}}$$

$$g_{N_p}(x(k+N_p)) = \frac{1}{2}x(k+N_p)^\top P_{N_p} x(k+N_p)$$

$$g_{N_p-1}(x(k+N_p-1)) \boxed{= \min_{u(k+N_p-1)} \left\{ \begin{array}{c} \frac{1}{2}x(k+N_p-1)^\top Q x(k+N_p-1) \\ +\frac{1}{2}u(k+N_p-1)^\top R u(k+N_p-1) \\ +\frac{1}{2}x(k+N_p)^\top P_{N_p} x(k+N_p) \end{array} \right\}}$$

APPLICATION: LTI, QUADRATIC COST

$$x(k+1) = Ax(k) + Bu(k)$$

$$g_{N_p-1}\big(x(k+N_p-1)\big) = \min_{u(k+N_p-1)} \left\{ \begin{array}{c} \frac{1}{2}x(k+N_p-1)^\intercal Q x(k+N_p-1) \\ +\frac{1}{2}u(k+N_p-1)^\intercal R u(k+N_p-1) \\ +\frac{1}{2}\big(Ax(k+N_p-1)+Bu(k+N_p-1)\big)^\intercal P_{N_p}\big(Ax(k+N_p-1)+Bu(k+N_p-1)\big) \end{array} \right\}$$

Show that $g_{N_p-1}\big(x(k+N_p-1)\big)$ can be rewritten as

$$g_{N_p-1}\big(x(k+N_p-1)\big) = x^\intercal(k+N_p-1)\mathrm{P}_{N_p-1}x(k+N_p-1)$$

TIPS : use the first order condition?

$$\boxed{\text{Give the expression of the recurrence defined by matrices } P_{N_p}}$$

$$x = x(k+N_p-1) \qquad g_{N_{p-1}} = \frac{1}{2}x^T Q x + \frac{1}{2}u^T R u + \frac{1}{2}\left((Ax+Bu)^T P_{N_p}(Ax+Bu)\right)$$
$$u = u(k+N_p-1)$$

$$\frac{\partial g}{\partial u} = Ru + B^T P_{N_p}(Ax+Bu) = 0$$

$$u = -\underbrace{(R + B^T P_{N_p} B)^{-1} B^T P_{N_p} A}\, x$$
$$= -K_{N_{p-1}}\, x$$

$$g_{N_{p-1}} = \frac{1}{2}x^T Q x + \frac{1}{2}x^T K^T R K x + \frac{1}{2}x^T (A-BK)^T P_{N_p}(A-BK)x$$
$$= \frac{1}{2}x^T\left[Q + K^T R K + A^T P_{N_p} A + K^T B^T P_{N_p} B K - 2A^T P_{N_p} B K\right]x$$
$$= \frac{1}{2}x^T\left[Q + A^T P_{N_p} A + \underline{K^T(R + B^T P_{N_p} B)K} - 2A^T P_{N_p} B K\right]x$$

$$+A^T P_{N_p} B K \qquad\qquad -2A^T P_{N_p} B K$$

$$= \frac{1}{2}x^T\left(Q + A^T P_{N_p} A - \cancel{A^T P_{N_p} B (R + B^T P_{N_p} B)^{-1} B^T P A}\right)x = \frac{1}{2}x^T P_{N_{p-1}}\, x$$

3.3 ON THE WAY …
TO INFINITY AND BEYOND

WHEN N_p GOES TO INFINITY

$$x(k+1) = Ax(k) + Bu(k), J^{\infty} = \frac{1}{2}\sum_{i=k}^{\infty}\left(x_i^{\intercal} Q x_i + u_i^{\intercal} R u_i\right),$$

Under good assumptions, it can be shown that:

- The sequence (P_N) converges to P, and the optimal control is

dLQR Control

$$u(k) = -Kx(k)$$
$$K = (R + B^{\intercal}PB)^{-1}B^{\intercal}PA$$
$$P = A^{\intercal}(P - PB(R + B^{\intercal}PB)^{-1}B^{\intercal}P)A + Q$$

Stabilizing state feedback!

Discrete Riccati Equation

$$P = A^{\intercal}(P - PB(R + B^{\intercal}PB)^{-1}B^{\intercal}P)A + Q$$

D-LQR

$$x(k+1) = Ax(k) + Bu(k)$$
$$J^{\infty} = \frac{1}{2}\sum_{i=k}^{\infty}(x_i^{\intercal}Qx_i + u_i^{\intercal}Ru_i)$$

dLQR Control

$$u(k) = -Kx(k)$$
$$K = (R + B^{\intercal}PB)^{-1}B^{\intercal}PA$$
$$P = A^{\intercal}(P - PB(R + B^{\intercal}PB)^{-1}B^{\intercal}P)A + Q$$

假设！

What are the assumptions?

- $R = R^{\intercal}, R \succ 0,\ Q = Q^{\intercal}, Q \succcurlyeq 0$ R： defini–positive
  Q: semi–defini positive
- (A, B) : stabilizable (controllable is sufficient)
- $Q = C^{\intercal}C$, (A, C) : observable

Riccati is a nice expression … but may be scaring too

- Good news : Matlab can compute it! *dlqr function*

3.4 CONCLUSIONS

CONCLUSION
LTI, QP FUNCTION, UNCONSTRAINED

Now, you have 3 ways to **stabilize** a system, using a state feedback

	Advantages	Drawbacks
Pole Placement	Quick computation	Controllable How to choose the poles? Stability margin not necessarly good
D-LQR approach	Convergence guaranteed Optimality Good Stabilty Margin	Stabilizable Linear sytem Computation …
MPC approach	Optimality Simple Concepts Many classes of systems	Stabilizable Stability margin not necessarly good Computation …

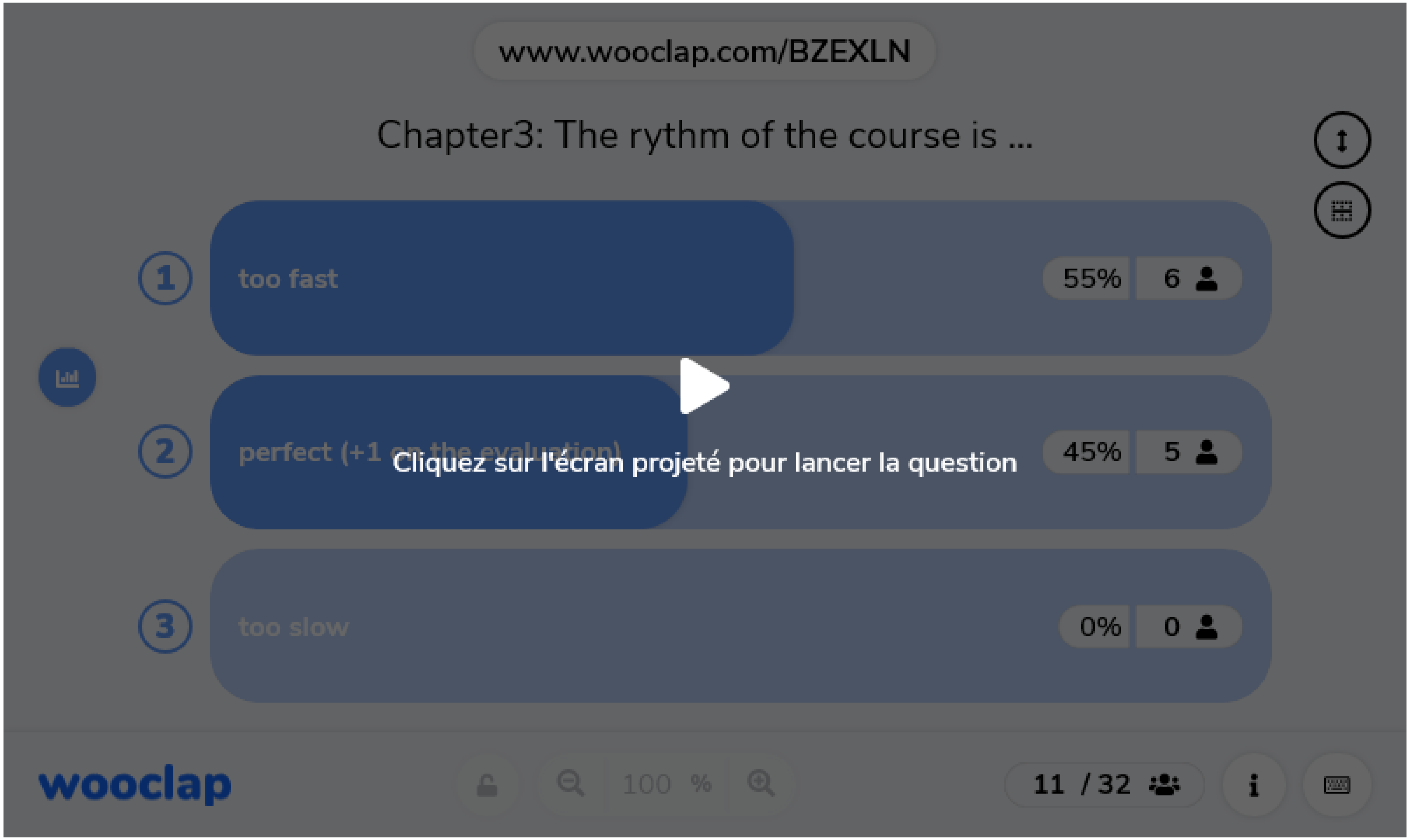
LQR (linear quadratic regulator)即线性二次型调节器，其对象是现代控制理论中以状态空间形式给出的线性系统，而目标函数为对象状态和控制输入的二次型函数。LQR最优设计是指设计出的状态反馈控制器 K要使二次型目标函数J 取最小值，而 K由权矩阵Q 与 R 唯一决定，故此 Q、 R 的选择尤为重要。LQR理论是现代控制理论中发展最早也最为成熟的一种状态空间设计法。特别可贵的是，LQR可得到状态线性反馈的最优控制规律，易于构成闭环最优控制。而且 Matlab 的应用为LQR 理论仿真提供了条件，更为我们实现稳、准、快的控制目标提供了方便。

对于线性系统的控制器设计问题，如果其性能指标是状态变量和(或)控制变量的二次型函数的积分，则这种动态系统的最优化问题称为线性系统二次型性能指标的最优控制问题，简称为线性二次型最优控制问题或线性二次问题。线性二次型问题的最优解可以写成统一的解析表达式和实现求解过程的规范化，并可简单地采用状态线性反馈控制律构成闭环最优控制系统，能够兼顾多项性能指标，因此得到特别的重视，为现代控制理论中发展较为成熟的一部分。

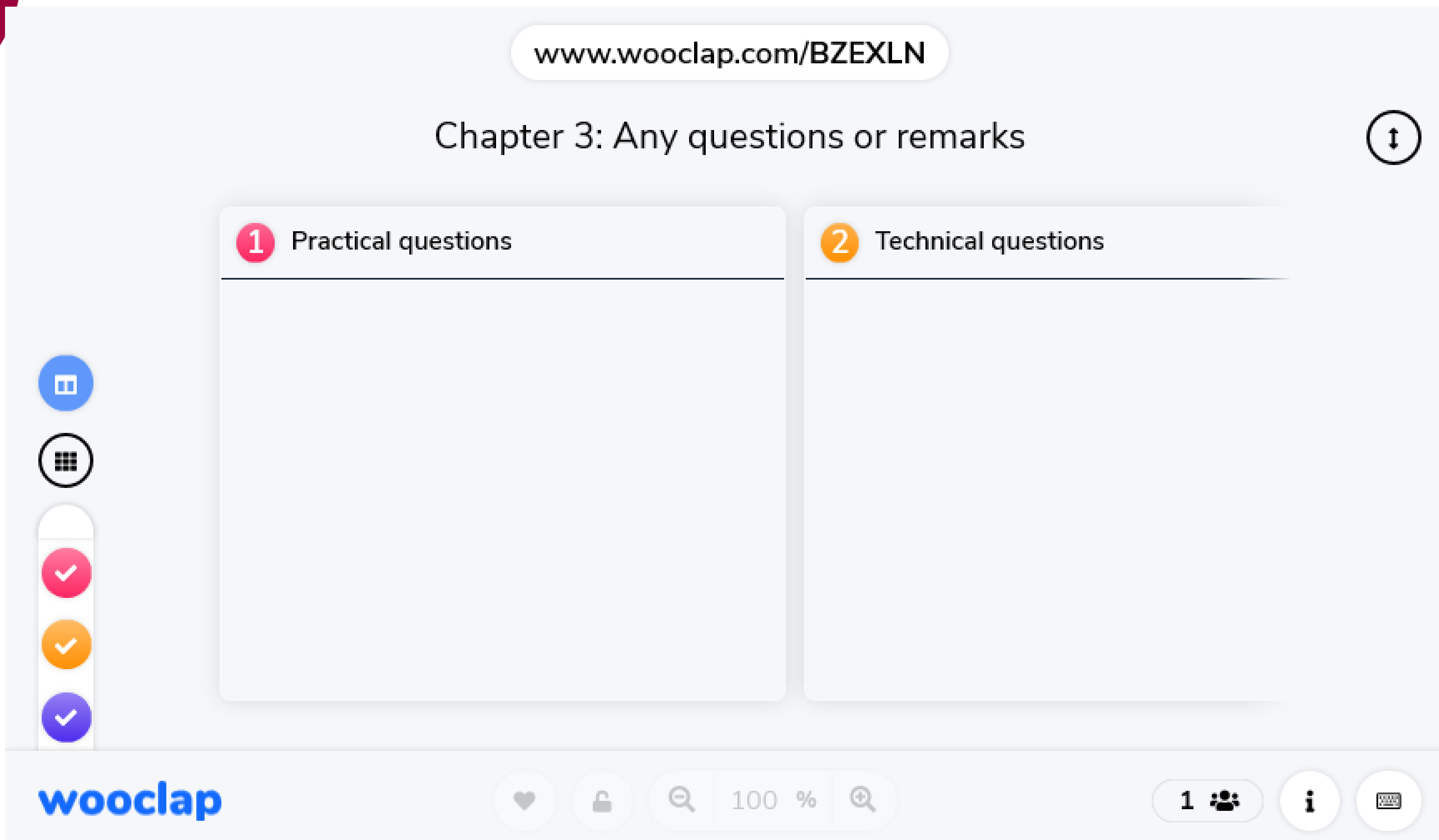
优点

LQR最优控制利用廉价成本可以使原系统达到较好的性能指标(事实也可以对不稳定的系统进行整定) ,而且方法简单便于实现 ,同时利用 Matlab 强大的功能体系容易对系统实现仿真。

2A-SG8-MPC-CHAPTER 3
RYTHM



2A-SG8-MPC-CHAPTER 3 QUESTIONS?



3.5 WORKING CLASS

YOUR TURN FOR EXAMPLE 3.1 ! WHICH IS NOW 3.4 !

Exercice 3.4

Let us consider the following discrete-time linear system

$$\left\{ x(k+1) = \begin{pmatrix} 1.1 & 2 \\ 0 & 0.95 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 0.0787 \end{pmatrix} u(k) \right.$$

and the criterion to optimize is

$$J^{N_p} = \sum_{i=1}^{N_p} \left(x^\intercal(k+i|k) \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} x(k+i|k) + \lambda u^2(k+i-1|k) \right)$$

1. We will pick up $\lambda = 0.1$. Propose a script that compare the MPC approach with the dlqr regulation.
2. Look at the eigenvalues of the closed-loop system for different values of N_p

THE « SMALL » CHANGE

Exercice 3.5

Let us consider the following discrete-time linear system

$$\left\{ x(k+1) = \begin{pmatrix} \mathbf{2.1} & 2 \\ 0 & 0.95 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 0.0787 \end{pmatrix} u(k) \right.$$

and the criterion to optimize is

$$J^{N_p} = \sum_{i=1}^{N_p} \left(x^\intercal(k+i|k) \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} x(k+i|k) + \lambda u^2(k+i-1|k) \right)$$

1. Let us fix $\lambda = 1$. Use the *dlqr* to stabilize the system.
2. Find a way to tune the MPC controller to stabilize the system. What problem occurs?

2A-SG8-MPC-CHAPTER 3 – THE BIG QUESTION: WHY SHOULD WE USE MPC ?

Many classes of Systems;

LQR控制器

维基百科，自由的百科全书

最优控制理論主要探討的是讓动力系统以在最小成本來運作，若系統動態可以用一組线性微分方程表示，而其成本為二次泛函，這類的問題稱為線性二次（LQ）問題。此類問題的解即為**線性二次調節器**（英語：linear–quadratic regulator），簡稱**LQR**。

LQR是回授控制器，方程式在後面會提到。LQR是LQG（線性二次高斯）問題解當中重要的一部份。而LQG問題和LQR問題都是控制理论中最基礎的問題之一。

目录

簡介

控制機器（例如飛機）的控制器，或是控制製程（例如化學反應）的控制器，可以進行最佳控制，方式是先設定成本函數，再由工程師設定加權，利用數學演算法來找到使成本函數最小化的設定值。成本函數一般會定義為主要量測量（例如飛行高度或是制程溫度）和理想值的偏差的和。演算法會設法調整參數，讓這些不希望出現的偏差降到最小。而控制量的大小本身也會包括在成本函數中。

LQR演算法減少了工程師為了讓控制器最佳化，而需付出的心力。不過工程師仍然要列出成本函數的相關參數，並且將結果和理想的設計目標比較。因此控制器的建構常會是迭代的，工程師在模擬過程中決定最佳控制器，再去調整參數讓結果更接近設計目標。

在本質上，LQR演算法是找尋合適狀態回授控制器的自動化方式。因此也常會有控制工程師用其他替代方式，例如全狀態回授（也稱為極點安置）的作法，此作法對控制器參數和控制器性能之間的關係比較明確。而LQR演算法的困難之處在找合適的加權因子，這也限制了以LQR控制器合成的相關應用。

有限時間長度，連續時間的LQR

方程式如下的連續時間線性系統，$t \in [t_0, t_1]$：

$$\dot{x} = Ax + Bu$$

其二次成本泛函為

$$J = x^T(t_1)F(t_1)x(t_1) + \int_{t_0}^{t_1} \left(x^T Qx + u^T Ru + 2x^T Nu\right) dt$$

其中F、Q和R都是正定矩陣。

可以讓成本最小化的回授控制律為

$$u = -Kx$$

其中K為

$$K = R^{-1}(B^T P(t) + N^T)$$

而P是連續時間Riccati方程的解：

$$A^T P(t) + P(t)A - (P(t)B + N)R^{-1}(B^T P(t) + N^T) + Q = -\dot{P}(t)$$

邊界條件如下

$$P(t_1) = F(t_1).$$

J_{min}的一階條件如下

(i) 狀態方程

$$\dot{x} = Ax + Bu$$

(ii) 協態方程

$$-\dot{\lambda} = Qx + Nu + A^T \lambda$$

(iii) 靜止方程

$$0 = Ru + N^T x + B^T \lambda$$

(iv) 邊界條件

$$x(t_0) = x_0$$

且 $\lambda(t_1) = F(t_1)x(t_1)$

無限時間長度，連續時間的LQR

考慮以下的連續時間線性系統

$$\dot{x} = Ax + Bu$$

其成本泛函為

$$J = \int_0^\infty \left(x^T Qx + u^T Ru + 2x^T Nu\right) dt$$

可以讓成本最小化的回授控制律為

$$u = -Kx$$

其中K定義為

$$K = R^{-1}(B^T P + N^T)$$

而P是代數Riccati方程的解

$$A^T P + PA - (PB + N)R^{-1}(B^T P + N^T) + Q = 0$$

也可以寫成下式

$$\mathcal{A}^T P + P\mathcal{A} - PBR^{-1}B^T P + \mathcal{Q} = 0$$

其中

$$\mathcal{A} = A - BR^{-1}N^T \qquad \mathcal{Q} = Q - NR^{-1}N^T$$

有限時間長度，離散時間的LQR

考慮離散時間的線性系統，定義如下 [1]

$$x_{k+1} = Ax_k + Bu_k$$

其性能指標為

$$J = x_N^T Q x_N + \sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k + 2x_k^T N u_k \right)$$

可以讓性能指標最小化的最佳控制序列為

$$u_k = -F_k x_k$$

其中

$$F_k = (R + B^T P_{k+1} B)^{-1}(B^T P_{k+1} A + N^T)$$

而P_k是由動態Riccati方程倒退時間佚代計算而得

$$P_{k-1} = A^T P_k A - (A^T P_k B + N)\left(R + B^T P_k B\right)^{-1}(B^T P_k A + N^T) + Q$$

從終端條件$P_N = Q$開始計算。注意u_N沒有定義，因為 x 是由$Ax_{N-1} + Bu_{N-1}$推導到其最終狀態x_N。

無限時間長度，離散時間的LQR

考慮離散時間的線性系統，定義如下

$$x_{k+1} = Ax_k + Bu_k$$

其性能指標為

$$J=\sum_{k=0}^{\infty}\left(x_k^T Q x_k + u_k^T R u_k + 2x_k^T N u_k\right)$$

可以讓性能指標最小化的最佳控制序列為

$$u_k = -F x_k$$

其中

$$F=(R+B^T P B)^{-1}(B^T P A + N^T)$$

而P是離散代數Riccati方程（DARE）的唯一正定解。

$$P=A^T P A-(A^T P B+N)\left(R+B^T P B\right)^{-1}(B^T P A+N^T)+Q.$$

可以寫成

$$P=\mathcal{A}^T P \mathcal{A}-\mathcal{A}^T P B\left(R+B^T P B\right)^{-1} B^T P \mathcal{A}+\mathcal{Q}$$

其中

$$\mathcal{A}=A-BR^{-1}N^T \qquad \mathcal{Q}=Q-NR^{-1}N^T.$$

而求解代數Riccati方程的一個方式是迭代計算有限時間的動態Riccati方程，直到所得的解收斂為止。

參考資料

1. Chow, Gregory C. Analysis and Control of Dynamic Economic Systems. Krieger Publ. Co. 1986. ISBN 0-89874-969-7.

   - Kwakernaak, Huibert & Sivan, Raphael. Linear Optimal Control Systems. First Edition. Wiley-Interscience. 1972. ISBN 0-471-51110-2.
   - Sontag, Eduardo. Mathematical Control Theory: Deterministic Finite Dimensional Systems. Second Edition. Springer. 1998. ISBN 0-387-98489-5.

外部連結

- MATLAB function for Linear Quadratic Regulator design (http://www.mathworks.com/help/toolbox/control/ref/lqr.html) （页面存档备份 (https://web.archive.org/web/20120824013759/http://www.mathworks.com/help/toolbox/control/ref/lqr.html)，存于互联网档案馆）
- Mathematica function for Linear Quadratic Regulator design (http://reference.wolfram.com/mathematica/ref/LQRegulatorGains.html) （页面存档备份 (https://web.archive.org/web/20131017135004/http://reference.wolfram.com/mathematica/ref/LQRegulatorGains.html)，存于互联网档案馆）