227-0216-00L: Computational Control FS2023

Dr. Michele Zaffalon
Bruker BioSpin AG

June 20, 2023

Unreviewed notes for Dr. Bolognani's class. Use at own risk!

Contents

Chapter 1

Dynamic Programming and LQR

Controlling a dynamic system consists in finding a vector of control inputs $\mathbf{u} = \{u_t\}_{t=0,1,\ldots}$ that minimize a cost function $J(\mathbf{u})$,

$$\min_{\mathbf{u}\in\mathcal{U}} J(\mathbf{u})$$

with the control input constrained to the space $\mathcal{U}$. This is generally a hard problem for the following reasons:

- the state space $\mathbf{u}$ is large;
- a fairly accurate model of the system is required. This can be obtained by first principles, by numerical simulation or can be a black box (oracle) that, given the control input $\mathbf{u}$, computes the system trajectory;
- the optimization problem is in general non-convex. For convex problems, which have a single (global) minimum[1], there exists efficient solvers.

From a practical point of view, the open-loop nature of the optimal control sequence makes it less reliable in the presence of disturbances and uncertainties.

[1]Not all functions with single minimum are convex: for instance

$$f(x) = -\frac{1}{1+x^2}$$

1.1 Bellman's Principle

The problem becomes tractable under the following conditions

- there exists a Markovian state that evolves according to

$$x_{t+1} = f_t(x_t, u_t).$$

  In other words, the complete state of the system is described by x_t which, together with u_t is the only thing needed to know for the future evolution;

- the initial condition x_0 is known;

- the cost is addictive over time

$$V(\mathbf{x}, \mathbf{u}) = \sum_t g_t(x_t, u_t).$$

Under these assumptions, Bellman's principle allows one to solve the optimization problem[2]

$$\begin{aligned} U(X_0) = \min_{\mathbf{u}} & \sum_{t=0}^{T} g_t(x_t, u_t) + g_T(x_T) \\ \text{subject to } & x_{t+1} = f_t(x_t, u_t) \\ & x_o = X_0 \\ & x_t \in \mathcal{X}_t \quad \forall t \\ & u_t \in \mathcal{U}_t \quad \forall t. \end{aligned} \tag{1.1}$$

1.1.1 Dynamic Programming

The problem is approached by iterating backwards from the last stage T: at T the problem is trivial since there is no control input to optimize for

$$V_T(x_T) = g_T(x_T).$$

[2]In the class notes the minimization is done over both states $\mathbf{x}$ and control inputs $\mathbf{u}$. In these notes, on the contrary I write the minimization only over $\mathbf{u}$, in line with [], since the states $\mathbf{x}$ are determined from the initial condition x_0 and the system dynmics. The explanation on Moodle justified the use of $\mathbf{x}$ from an implementation point of view only.

At time $T-1$, we need to minimize

$$V_{T-1}(x_{T-1}) = \begin{cases} \min\limits_{u_{T-1}} g_{T-1}(x_{T-1}, u_{T-1}) + V_T(x_T) \\ \text{subject to } x_T = f_{T-1}(x_{T-1}, u_{T-1}) \end{cases}$$
$$= \min_{u_{T-1}} g_{T-1}(x_{T-1}, u_{T-1}) + V_T(f_{T-1}(x_{T-1}, u_{T-1})$$

and at a generic time t

$$V_t(x_t) = \min_{u_t} \left\{ g_t(x_t, u_t) + V_{t+1}(f_t(x_t, u_t)) \right\}. \tag{1.2}$$

In other words, the problem is decomposed into stage problems that can be solved by backward induction. The problem is computationally easy to solve if

- the decisions space $\mathcal{U}$ is convex;
- $g_t(\cdot, u_t)$ is convex in u_t. Note that we do not request g_t to be convex in x_t because x_t is a parameter of the problem;
- $V_{t+1}(f_t(\cdot, u_t))$ is convex in u_t. A sufficient condition is that V_{t+1} is convex in x_{t+1} and f_t is convex in u_t.

In practice the problem is tractable only for quadratic cost g and linear dynamics f.

1.2 LQR

LQR problems have quadratic cost functions[3], linear dynamics and no constraints

$$\begin{aligned} &\min_{\mathbf{u}} \sum_{t=0}^{T-1} \left(||x_t||_Q^2 + ||u_t||_R^2 \right) + ||x_T||_S^2, \quad Q, S \succeq 0,\ R \succ 0 \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t. \end{aligned} \tag{1.3}$$

[3]A more general quadratic cost includes the cross terms in x and u

$$\begin{bmatrix} x^\top & u^\top \end{bmatrix} \begin{bmatrix} Q & N \\ N^\top & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} = x^\top Q x + u^\top R u + 2x^\top N u.$$

This quadratic cost function can represent any linear feedback controller $u = -\bar{K}x$ when letting $Q = \bar{K}^\top \bar{K}$, $R = 1$ and $N = \bar{K}^\top$. To see why, we look for a controller $u = -Kx$ that minimizes the expression

$$x^\top Q x + u^\top R u + 2x^\top N u = x^\top \left(\bar{K}^\top \bar{K} + K^\top K - 2\bar{K}^\top K \right) x = x^\top \left(\bar{K} - K \right)^2 x$$

this occurs at $K = \bar{K}$.

One imposes a *strictly* positive cost on the control inputs, $R \succ 0$ to reflect the case that is relevant in practice where there is a cost associated with actuating the system.

Proof. The LQR can be solved analytically and the solution is of the form

$$V_t(x) = x^\top P_t x, \quad P_t \succeq 0. \tag{1.4}$$

We show this by induction starting from $t = T$ and recursing backwards.

The initial case is satisfied, because $V_T(x) = x^\top S x$ is quadratic and $P_T = S$. At time t, assuming that $V_{t+1}(x) = x^\top P_{t+1} x$, we have $V_t(x) = \min_u J_t(x,u) = J_t(x,u^\star)$ where

$$\begin{aligned} J_t(x,u) &= x^\top Q x + u^\top R u + V_{t+1}(Ax + Bu) \\ &= x^\top \left(Q + A^\top P_{t+1} A\right) x + u^\top \left(R + B^\top P_{t+1} B\right) u + 2u^\top B^\top P_{t+1} A x. \end{aligned}$$

To find the minimum $u^\star$ we set its gradient to zero, $\nabla_u J(x,u) = 0$

$$\begin{aligned} &\left(R + B^\top P_{t+1} B\right) u^\star + B^\top P_{t+1} A x = 0 \\ &\to u^\star = -\underbrace{(R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A}_{K_t} x. \end{aligned} \tag{1.5}$$

This solution exists since $R + B^\top P_{t+1} B$ being a strictly positive $m \times m$ definite matrix can be inverted. We need to prove that $V_t(x) = x^\top P_t x$. We plug the optimal control $u^\star$ into the expression for $V_t(x) = J_t(x,u^\star)$ to obtain

$$V_t(x) = x^\top \left(Q + A^\top P_{t+1} A\right) x - x^\top A^\top P_{t+1} B \left(R + B^\top P_{t+1} B\right)^{-1} B^\top P_{t+1} A x$$

where

$$P_t = Q + A^\top P_{t+1} A - A^\top P_{t+1} B \left(R + B^\top P_{t+1} B\right)^{-1} B^\top P_{t+1} A. \tag{1.6}$$

$\square$

LQR finds an optimal control sequence $\mathbf{u}^\star = \{u_0^\star, u_1^\star, \ldots, u_{T-1}^\star\}$ computating the matrices K_t by backward recursion, using eq. (1.6) and eq. (1.5) followed by forward integration of the system dynamics from the initial state x_0 to obtain the next state x_t and the next control input $u_t^\star = -K_t x_t$.

Note that the optimal control sequence is a feedback law, $u_t = -K_t x_t$. In case of state perturbation, the system evolves differently from the predicted state, but applying the input $u_t = -K_t\tilde{x}_t$ on the measured state $\tilde{x}_t$ guarantees optimality because by Bellman's principle, the computed sequence is optimal from any starting point and onwards. Contrast this with model mismatch: in this case the feedback law using K_t is not optimal because it relies on the matrices A and B being an accurate representation of the system.

1.2.1 Infinite Horizon LQR

There are situations where one wants to maintain a stationary state. This is captured by making the horizon infinite,

$$\begin{aligned} &\min_{\mathbf{u}} \sum_{t=0}^{\infty} \left(||x_t||_Q^2 + ||u_t||_R^2 \right), \quad Q \succeq 0, R \succ 0 \\ &\text{subject to } x_{t+1} = Ax_t + Bu_u, \quad x_0 = X_0. \end{aligned} \tag{1.7}$$

so that the problem becomes time-invariant and only one feedback matrix K_∞ is required.

The convergence of eq. (1.7) is guaranteed by the following observation: if the system is stabilizable[4], there is an input sequence that yields a finite cost. To see why, let $u_t = -Kx_t$ a linear feedback such that $A - BK$ has eigenvalues inside the unit circle and $x_t = (A - BK)^t X_0$. The cost to go

$$\begin{aligned} V(X_0) &= \sum_{t=0}^{\infty} x_t^\top Q x_t \; + x_t^\top K^\top R K x_t \\ &= X_0^\top \sum_{t=0}^{\infty} (A - BK)^{t\top} \left(Q + K^\top R K \right) (A - BK)^t X_0 \end{aligned}$$

is a geometric converging series.

Since the problem is time-invariant, the feedback law is computed by solving the time-invariant ARE with the subindex t dropped in eq. (1.6)

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

using *e.g.* an ARE solver or analytically. Note that AREs have multiple solutions and for a controllable system, the *optimal* solution is the matrix P_∞ with the minimum cone. A solver will pick the correct solution. Alternatively, one can iterate eq. (1.6) until convergence. It can be proven that the iteration converges to the solution with the smallest cone.

[4]A system is stabilizable if all uncontrollable state variables can be made to have stable dynamics.

The optimal K_∞ does however not stabilize the system unless the uncontrollable modes are stable and the controllable unstable modes[5] are weighted in Q:

Theorem 1. *The feedback K_∞ stabilizes the system if and only if (A, B) is stabilizable and $(Q^{1/2}, A)$ is observable.*

The existance of a stabilizing controller K is guaranteed by the fact that (A, B) is stabilizable. Since $(Q^{1/2}, A)$ is observable, the unstable states of A appear at the output $y = Q^{1/2}x$, which also enters the objective function as the state dependent term

$$||y||^2 = \left(x^\top Q^{1/2}\right)\left(Q^{1/2}x\ \ \right) = x^\top Qx.$$

The convergence of the infinite horizon cost V implies the convergence of y and by stabilizability, the convergence of x.

1.2.2 Stability of the Optimal Controller

If the system has some unobservable states, the optimal control law K_∞ does not stabilize the system, because optimality of the controller is with respect to the cost function. Consider the unstable system

$$x^+ = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} x + u, \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad u = -Kx$$

where the diagonal A matrix has eigenvalues outside of the unit circle, $\lambda = 2, 3$. The system can be controlled by an appropriate choice of K which cannot not be optimal because acting on the second state incurs a cost, since $R_{22} \neq 0$, while there is no cost associated with letting the second state diverge because $Q_{22} = 0$: the optimal solution will therefore yield zero control input of the second state.

However, even if a mode is not stable and not weighted in the cost function, it can still be stabilized with a controller that is suboptimal, by solving the ARE and selecting the *largest* cone solution P_S. It can be proven that

- there is at most one stabilizing solution P_S;
- of all solutions P to the ARE, P_S is the one with the largest cone;

[5]It looks like $(Q^{1/2}, A)$ being detectable should be sufficient, but theorem 8.1 in "Predictive Control for linear and hybrid systems" by Borrelli, Bemporad and Morari requires observability of the pair.

- if P_∞ is stabilizing, it is the only solution to the ARE;
- P_S is the limit of the backward induction initialized at any $P_0 \neq 0$;
- P_S is the optimal *stabilizing* feedback solution.

For the example above, the Ricatti equation gives a diagonal matrix whose entries and solutions are

$$\begin{aligned} 1 + 3p_1 - \frac{4p_1^2}{1+p_1} &= 0 \ \rightarrow p_1 = 2 \pm \sqrt{5} \\ 8p_2 - \frac{9p_2^2}{1+p_2} &= 0 \ \rightarrow p_2 = 0, 8 \end{aligned}$$

and the optimal *stabilizing* feedback solution is

$$P_S = \begin{bmatrix} 2+\sqrt{5} & 0 \\ 0 & 8 \end{bmatrix}, \quad K_S = 2 \begin{bmatrix} \frac{2+\sqrt{5}}{3+\sqrt{5}} & 0 \\ 0 & \frac{4}{3} \end{bmatrix}.$$

Chapter 2

Model Predictive Control

Model Predictive Control (MPC), also referred to as receding horizon control, is a process control technique that optimizes a cost function by using a process model for the state evolution with the additional requirements that the control input $\mathbf{u} = \{u_0, u_1, \ldots, u_{K-1}\}$ and the states $\mathbf{x} = \{x_0, x_1, \ldots, x_{K-1}, x_K\}$ also satisfy a set of contraints[1]:

$$\begin{aligned} &\min_{\mathbf{u}} \sum_{k=0}^{K-1} g_k(x_k, u_k) + g_K(x_K), \\ \text{subject to } & x_{k+1} = f(x_k, u_k) \\ & x_0 = X_0 \\ & x_k \in \mathcal{X}_k \\ & u_k \in \mathcal{U}_k \end{aligned} \tag{2.1}$$

where X_0 is the initial state and g_k is a non-negative cost function such as a quadratic one

$$g_k(x, u) = ||x||_{Q_k} + ||u||_{R_k} \tag{2.2}$$

In the presence of constraints, one cannot find the optimal solution by setting the gradients $\nabla_{u_k} V(\mathbf{x}, \mathbf{u})$ of the cost-to-go

$$V(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{K-1} g_k(x_k, u_k) + g_K(x_K) \tag{2.3}$$

[1]Input constraints often represent physical limits: if the controller does not respect the input constraints, the plant will enforce them. This is in contrast with state constraints that are normally desirable. For this reason, MPC are often setup as optimization problems using hard constraints for the input and some sort of soft constraints for the states [2].

to zero as it was the case for the LQR problem. The optimal control input sequence $\mathbf{u}^\star = \{u_0^\star, u_1^\star, \dots, u_{K-1}^\star\}$ and the optimal trajectory $\mathbf{x}^\star = \{x_0^\star, x_1^\star, \dots, x_K^\star\}$ can however be found if the optimization problem is feasible. In open-loop control, the whole optimal sequence of input control inputs $\mathbf{u}^\star$ is applied and the system evolves to reach the terminal state x_K after K steps.

In case of model mismatch or state perturbation however, the system will deviate increasingly more from the predicted trajectory $\mathbf{x}^\star$. If a measurement of the system state is available, it makes sense to find a new optimal control input vector using the measured state as the initial point for the optimization. (Question: if this is the case, then why not optimizing only the tail such that also in closed loop the trajectory is completed after K steps?)

Closed-loop control therefore applies the following strategy: at time step i with the system in state x_i, compute the optimal control input $\mathbf{u}_i^\star = \{u_{i,0}^\star, u_{i,1}^\star \dots, u_{i,K-1}^\star\}$, apply the first element of it, $u_{i,0}^\star$, and discard the remaining elements. At the following time step $i+1$ and with the system having evolved into state x_{i+1}, solve the optimization from the initial state x_{i+1} to find the optimal control input $\mathbf{u}_{i+1}^\star$, apply the first element $u_{i+1,0}^\star$ and discard the remaining elements. This process is repeated.

The closed-loop approach permits us to solve the optimization problem with an horizon of length K in a receding way, by restarting the control process from the current state. This is possible since the system is Markovian and the current state is all one needs to compute the optimal control for the future.

If the model closely represents the system and in absence of perturbations, the closed-loop approach can be restarted using the predicted state $x_{k+1} = f(x_k, u_k)$; in this case the optimal state starting for $i+1$ concides with that evolved from $x_{i,0}^\star$ and under the optimal control $u_{i,0}^\star$: $x_{i+1,0}^\star = f(x_{i,0}^\star, u_{i,0}^\star)$.

Either way, closed-loop MPC solves repeatedly the exact same problem eq. (2.1), parametrised by the initial state $x_0 = X_0$. Since the problem is the same, the optimal solution $\mathbf{u}^\star$ and in particular its first element $u_0^\star$ are independent from the time index i and the index will be dropped in the following.

In general there is no guarantee that the map $x \to u_0^\star(x)$ is a continuous function even under the assumption of the continuity of f and g_k; the cost-to-go $J(\mathbf{x}, \mathbf{u})$ is instead continuous in the parameters $\mathbf{x}$ and $\mathbf{u}$. Moreover, under compactness assumptions on the constraints, the parametric optimization problem has a solution when it is feasible (I do not understand this sentence).

From a computational point of view, the problem eq. (2.1) can be efficiently solved if it is convex, which happens for instance if the stage cost g_k is a convex function, the spaces $\mathcal{X}_k$ and $\mathcal{U}_k$ are convex and the dynamics is linear $f(x_k, u_k) = Ax_k + Bu_k$.

To summarize: MPC is a static[2], nonlinear, time-invariant feedback[3] control-law.

2.0.1 Does MPC Realize the Optimal Control Problem Solution?

Despite MPC solving at each time step i the optimal control problem eq. (2.1), the MPC trajectory does not realize the optimal trajectory $\mathbf{u}_i^\star$, because MPC only applies the first control input $u_0^\star$ and discards the rest. Consider the pathological problem

$$\begin{gathered}\min_{\mathbf{u}} \sum_{k=0}^{K-1} \frac{1}{k} ||u_k||^2 + ||x(K)||^2 \\ \text{subject to } x_{k+1} = x_k + u_k.\end{gathered}$$

The optimization finds a solution where the first control input $u_0^\star$ must be zero or the cost explodes and delays the work to minimize the cost to go to a later time. By applying only the first control input, MPC will remain stuck in its initial state and the optimal control solution is never realized. This is so also in absence of disturbance and model mismatch.

From this example, it is clear that MPC is a way to compute a good controller but in no way a robust optimal controller[4].

2.0.2 MPC and Finite-Time LQR

Given the finite-time LQR problem eq. (1.3), we have seen that the LQR solution computes the matrices $\{P_K, P_{K-1}, \dots P_1\}$ by backward induction and constructs from them the linear feedback law $u = -\Gamma_k x$:

$$u_0^{\text{LQR}} = -\Gamma_0 x_0, \quad u_1^{\text{LQR}} = -\Gamma_1 \left(Ax_0 + Bu_0^{\text{LQR}}\right), \quad \dots$$

[2]A static controller has no memory of the past and depends only on the current statte x_k:

$$u_k = \varphi_k(x_k)$$

whereas a dynamic controller has memory, such as in the case of a PI controller

$$\xi_{k+1} = \xi_k + \varphi_k(x_k), \quad u_k = \psi_k(x_k) + \xi_k.$$

[3]Because f does not depend on k.

[4]Needs explanation: what is a robust optimal controller?

Compare this with the MPC approach[5]: the optimal control solution is the same as that of the LQR, since MPC solves the same problem, and finds the same sequence $\{P_K, P_{K-1}, \dots P_1\}$. MPC however applies the linear *time-invariant* feedback law $x = -\Gamma_0 x$:

$$u_0^{\mathrm{MPC}} = -\Gamma_0 x_0, \quad u_1^{\mathrm{MPC}} = -\Gamma_0 \left(Ax_0 + Bu_0^{\mathrm{MPC}}\right), \quad \dots$$

In the case of infinite horizon problems, the feedback law is the same, since the feedback law becomes time invariant also for the LQR problem.

2.1 Stability

MPC is a principled way to design static, time-invariant, non-linear feedback control law. While local stability can be analyzed by linearizing the control law around the equilibrium and using the techniques from Control Systems (Nyquist, Bode,...), the tool to investigate global stability of non-linear systems is the Lyapunov theorem.

2.1.1 Lyapunov Stability

Definition 2.1.1. The point $x = 0$ is *stable* for the dynamics $x_{k+1} = f(x_k)$ if a small perturbation of the state perturbs the subsequent state trajectory in a continuous manner.

In mathematical terms,

$$\forall \epsilon > 0, \ \exists \delta > 0 \text{ such that } ||x_0|| < \delta \to ||x_k|| < \epsilon \ \forall k \geq 0. \tag{2.4}$$

In other words, given a ball of radius ϵ, any trajectory starting from within a ball of radius δ will never leave the larger ball with radius ϵ.

An equilibrium is called *asymptotically stable* if it is stable and $\lim_{k\to\infty} ||x_k|| = 0$.

Theorem 2 (Lyapunov theorem)**.** *Given a discrete system with dynamics $x_{k+1} = f(x_k)$ and equilibrium $x = 0$ (?), if $W(x)$ is a real-valued function such that*

$$W(0) = 0 \textit{ and } W(x) > 0 \quad \forall x \neq 0$$
$$W(f(x)) < W(x) \quad \forall x \neq 0$$

then the point $x = 0$ is asymptotically stable.

[5]Why are we comparing a finite horizon finite simulation time LQR with MPC that has an infinite *simulation* time? I did not get the point, aside the obvious one that infinite horizon LQR and MPC give the same result. Either clarify or remove this slide.

A survey of formulation of Lyapunov theory for discrete systems is available at `https://arxiv.org/abs/1809.05289`.

2.1.2 Stability of MPC

To prove asymptotically stability of the closed loop problem[6], we need to find a Lyapunov function $W(x)$ that satisfies theorem 2, for the equilibrium point $(x_S, u_S) = (0, 0)$.

We consider first the simpler case of stability for the infinite horizon unconstrained problem. The presence of constraints does not alter the validity of the proof. The function

$$W(x) = \min_{\mathbf{u}} \sum_{k=0}^{\infty} g_k(x_k, u_k).$$

is a candidate for the Lyapunov function provided that 1. $W(x)$ is only zero at the target equilibrium (such as when g_k is a quadratic objective function) and 2. that it decreases along trajectories of the system. This is the case because the MPC controller finds $(\mathbf{x}^\star, \mathbf{u}^\star)$ that minimizes the problem

$$V_i^{\infty}(\mathbf{x}, \mathbf{u}) \equiv \sum_{k=i}^{\infty} g_k(x_k, u_k).$$

By Bellman's principle, the optimal trajectory minimizes also the "tail" cost starting at the subsequent time $i + 1$

$$V_{i+1}^{\infty}(\mathbf{x}, \mathbf{u}) = \sum_{k=i+1}^{\infty} g_k(x_k, u_k)$$

and therefore we have that

$$W(x_{i+1}) = W(x_i) - g_i(x_i, u_0^\star(x_i)) < W(x_i).$$

This poses the restriction that g_k must be a strictly positive quantity: if a state $\bar{\boldsymbol{x}}$ is not weighted in the cost function and $g_k(\bar{\boldsymbol{x}}) = 0$, then $W(x)$ defined above is not strictly decreasing and would not be a Lyapunov function for the asymptotic stability test.

For the finite receding MPC problem with horizon length K, we only consider the simpler problem of stability for a stage cost g that is independent

[6]Note that the open loop is trivially asymptotically stable because it consists of a sequence of finite length K.

from the index k. The trick above does not apply because at $i+1$, the new cost to go includes the term $g(x_{i+K})$ and excludes $g(x_i)$

$$V_{i+1}^K(\mathbf{u}, \mathbf{x}) = \sum_{k=i+1}^{i+K} g(x_k, u_k) = V_i^K(\mathbf{u}, \mathbf{x}) + g(x_{i+K}, u_{i+K}) - g(x_i, u_i)$$

and there is no guarantee that the requirement

$$W(x_{i+1}) = \min_{\mathbf{u}} V_{i+1}^K(\mathbf{u}, \mathbf{x}') < \min_{\mathbf{u}'} V_i^K(\mathbf{u}') = W(x_i)$$

is satisfied.

We can however construct a new optimization problem equal to the original one with the additional constraint $x_K = 0$ on last state x_K

$$\begin{aligned} W(x) = \min_{\mathbf{u}} \quad & \sum_{k=0}^{K-1} g(x_k, u_k) \\ \text{subject to} \quad & x_{k+1} = f(x_k, u_k) \\ & x_0 = x,\ x_K = 0 \end{aligned} \tag{2.5}$$

based on the intuition that the final state will be close to zero because one is trying to stabilize the system. This is a Lyapunov function if we prove that this is decreasing along a trajectory.

Assume we found the optimal solution $\mathbf{u}_i^\star$ to the modified problem for time step i from the initial state $x_{i,0} \neq 0$ that is not already the equilibrium. We construct the suboptimal control input $\mathbf{u}_{i+1}$ obtained by taking the optimal solution $\mathbf{u}_i^\star$, removing $u_{i,0}^\star$ and adding as the last input $u_{i+1,K-1} = 0$. Using $\mathbf{u}_{i+1}$, the trajectory $\mathbf{x}_{i+1}$ coincides with $\mathbf{x}_i^\star$: in particular, $x_{i+1,K-1} = x_{i,K}^\star = x_K = 0$ because of the constraint imposed in eq. (2.5). The final state[7]

$$x_{i+1,K} = Ax_{i+1,K-1} + Bu_{i+1,K-1} = 0.$$

$\mathbf{u}_{i+1}$ has a lower associated cost than $\mathbf{u}_i^\star$ because we have removed the strictly positive $g_k(x_{i,0}^\star, u_{i,0}^\star)$, since $x_{i,0} \neq 0$; conversely, the suboptimal $\mathbf{u}_{i+1}$

[7]This does not seem to be relevant for the proof.

has a higher (or equal) cost than the optimal solution $\mathbf{u}_{i+1}^\star$:

$$\begin{aligned}
&W(x_i) \\
&= V_i^K(\mathbf{x}_i^\star, \mathbf{u}_i^\star) && \text{definition of } W \\
&= V_{i+1}^K(\mathbf{x}_{i+1}, \mathbf{u}_{i+1}) + \underbrace{g(x_{i,0}^\star, u_{i,0}^\star)}_{>0} - \underbrace{g(x_{i+1,K-1}, u_{i+1,K-1})}_{=0} && \text{definition of } V_{i+1}^K \\
&> V_{i+1}^K(\mathbf{x}_{i+1}, \mathbf{u}_{i+1}) && \text{drop positive terms} \\
&\geq V_{i+1}^K(\mathbf{x}_{i+1}^\star, \mathbf{u}_{i+1}^\star) && \text{optimality of } \mathbf{u}_{i+1}^\star \\
&= W(x_{i+1}) && \text{definition of } W
\end{aligned}$$

This proves that $W(\cdot)$ is a decreasing function on a trajectory.

2.2 Implementation of MPC

An optimal MPC trajectory can be computing using an offline or an online approach.

2.2.1 Offline Computation of the Optimal Control Input

The offline method computes the *function* $u_0^\star(\cdot)$. This may require gridding of the solution and is a complex problem because of the exponential number of regions where the solution must be partitioned. To illustrate the approach, in the following section, we will solve a toy problem with time horizon $K = 1$ using the KKT condition.

Short Introduction to the KKT Condition

This section is a minimal review of the KKT condition. The minimization problem

$$\begin{gathered}\min_x \ f(x) \\ \text{subject to} \ \ g(x) \leq 0\end{gathered}$$

is equivalent to

$$\begin{cases}\nabla f(x) + \sum_i \mu_i \nabla g_i(x) = 0 \\ g(x) \leq 0 \\ \mu_i g_i(x) = 0 \ \forall i\end{cases}$$

with $\mu_i \geq 0$ and provided f and g are differentiable.

The condition breaks down into the following two cases:

- the minimum $\mathbf{x}^\star$ is inside the feasible domain as detemined by $g_i(\mathbf{x}^\star) < 0$, in which case $\mu_i = 0$ and $\nabla f(\mathbf{x}^\star) = 0$. This is the usual case where the minimum is found by setting the gradient to zero and the constraints have no effect, or
- the minimum occur at the boundary $g_i(\mathbf{x}^\star) = 0$, $\mu_i \neq 0$ and the two vectors $\nabla f(\mathbf{x}^\star)$ and $\sum_i \nabla g_i(\mathbf{x}^\star)$ are parallel.

The behaviour is qualitatively different whether the minimum is inside or at the boundary.

Offline computation: a Simple Minimization Problem

Let us consider the following toy example:

$$\begin{aligned} \min_{u_0, x_1} \; & u_0^2 + x_1^2 \\ \text{subject to } \; & x_1 = x_0 + u_0 \\ & x_1 \leq 1. \end{aligned}$$

For such simple cases, the KKT condition is all one needs to find minimizers. Eliminating x_1 gives the minimization problem

$$\begin{aligned} \min_{u_0} \; & f(u_0) \\ \text{subject to } \; & g(u_0) \leq 0 \end{aligned}$$

where $f(u_0) = u_0^2 + (x_0 + u_0)^2$, $g(u_0) = x_0 + u_0 - 1$ and $\nabla_{u_0} f(u_0) + \mu \nabla_{u_0} g(u_0) = 4u_0 + 2x_0 + \mu$. We consider the two cases

- $\mu = 0$: we have

$$\begin{aligned} & 4u_0 + 2x_0 = 0 \rightarrow u_0 = -\frac{x_0}{2} \\ & x_0 + u_0 - 1 < 0 \rightarrow x_0 < 2. \end{aligned}$$

- $\mu > 0$: from $\mu g(x) = 0$ the minimum is at the boundary $g(x) = 0$ and $4u_0 + 2x_0 + \mu > 4u_0 + 2x_0$ since $\mu > 0$. We have

$$\begin{aligned} & x_0 + u_0 - 1 = 0 \rightarrow u_0 = 1 - x_0 \\ & 4u_0 + 2x_0 = 4(1 - x_0) + 2x_0 > 0 \rightarrow x_0 > 2. \end{aligned}$$

Note that the solution is continuous at $x_0 = 2$: we have therefore the solution

$$\begin{cases} u_0 = -\frac{x_0}{2} & \text{for } x_0 < 2 \\ u_0 = 1 - x_0 & \text{for } x_0 \geq 2. \end{cases}$$

In this example, each constraint defines two (affine) regions and for each region there is a different control action. If there were n affine constraints, there would be at most 2^n regions, according to the way the constraints overlap. For a large number of constraints, the problem becomes intractables also for offline computation. In practice, one problem to solve is to figure out where the current state is in the state space.

2.2.2 Online Computation of the Optimal Control Input

Online computation computes the *value* $u_0^\star(\cdot)$ at every iteration and therefore it is typically applied to processes that have constraints and mainly used in slow dynamics.

2.2.3 Other Practical Implementations Aspects

Real world control problems often have an infinite horizon: they are computationally impossible to solve, unless the system has special properties. The finite horizon receding MPC is a computationally tractable approximation to the infinite horizon problem, the approximation being good provided K is chosen large enough that the interesting dynamics of the original problem has a time-scale shorter than K.

The "tail" of the infinite horizon problem is taken care by the MPC formulation in one of the following ways:

- by adding a final cost or constraints conditions on the final state: since the point $(x, u) = (0, 0)$ is an equilibrium, it is reasonable to impose the constraint $x_K = 0$ on the last state;
- by imposing that x_K live in a small space $\mathcal{X}_K$: if there is a region of the state space sufficiently close to equilibrium that is feasible, the trajectory starting at time $i + 1$ will keep the system feasible and hopefully easy to control;
- by approximating the tail of the infinite horizon problem from element $K + 1$ onwards by a final cost $g_K(x_K)$ that is large if the terminal state x_K is far from the equilibrium.

We have assumed feasibility (needs to be define or/and give an example of problem that becomes unfeasible) of the MPC problem which may break in the absence of a zero terminal constraint.

2.3 Steady-State Selection

Until now we have assumed we want to regulate at the steady-state $(x_s, u_s) = (0, 0)$. There are situations where we want the plant to operate at a working point $x_s \neq 0$ and to allow for a constant input u_s.

If the desired equilibrium (x_s, u_s) is known in advance, no changes to the MPC regulator are necessary except for replacing the stage cost $g_k(x, u)$ by $g_k(x - x_s, u - u_s)$ so that its minimum is at (x_s, u_s).

2.3.1 Steady-State Selection from Specifications

Sometimes the problem specifies a working point $(x_{\text{spec}}, u_{\text{spec}})$ so selected because it is safe and/or economic to operate. This working point may not be a valid state, *i.e* it does not satisfy $x_{\text{spec}} = f(x_{\text{spec}}, u_{\text{spec}})$. To find a valid steady-state, one solves offline the optimization problem

$$\begin{aligned} \min_{x_s, u_s} & ||x_s - x_{\text{spec}}||^2_{Q_s} + ||u_s - u_{\text{spec}}||^2_{R_s} \\ \text{subject to } & x_s = f(x_s, u_s) \\ & x_s \in \mathcal{X}, \; u_s \in \mathcal{U}. \end{aligned}$$

2.3.2 Incremental Formulation of MPC

A common application of feedback is to compensate for unmeasured disturbances to the system such that the selected controlled variables approach their setpoint asymptotically and without offset, even in the case of perfect plant model $x^+ = f(x, u)$.

a step perturbation cannot be rejected while at the same time reaching the equilibrium point $x = 0$ without the presence of an integrator in the plant.

The incremental formulation of MPC uses input variations Δu_k instead of the absolute input u_k and augments the state space by adding the control input $\begin{bmatrix} x & u \end{bmatrix}^\top$

In the truck platoon example, keeping the distance to the previous truck constant cannot be relied on only by knowing the (non-linear) relation between gas pedal angle and truck speed, even in presence of feedback. Instead the gas pedal is adjusted to compensate for variations of parameters (engine efficiency, friction, wind gusts. . .): the controller adjusts Δu_k which is integrated to generate the control input u_k that is then fed to the plant. This has also the practical advantage in that it is easier to implement input weights penalizing *changes* in the input signal.

An integrator may already be present in the plant. For example in drug administration, the body integrates the amount of drug (while at the same time slowly using it). (Should it one in this case also measure the integrated u_k since now this is part of the augmented state x_k, u_k?)

2.3.3 Disturbance Rejection in MPC

Chapter 3

Economic MPC

Many modern control problems present complex performance metrics, such as economic aspects, regulatory specifications and use of resources. Consider an iron furnace, where the inputs are its temperature, the air flow, the inflow of iron ore, limestone and coke, and the products includes the iron yield, its quality, the CO_2 emission and the amount of slag. The stage cost $\ell(x, u)$ represents economic losses, energy use, cost of materials, yield.... $\ell(x, u)$ is a continuous lower-bounded (otherwise the problem is not well defined) function.

One approach is to split the problem into the offline computation of the steady-state, *e.g.* the optimal furnace temperature, from the economic cost to go

$$\begin{aligned} \min_{x_s, u_s} \; & \ell(x_s, u_s) \\ \text{subject to } & x_s = f(x_s, u_s) \\ & x \in \mathcal{X}, \; u \in \mathcal{U} \end{aligned}$$

and an online MPC to track the steady state (x_s, u_s) using a quadratic cost function $g_k(x_k - x_s, u_k - u_s)$ that has the minimum at (x_s, u_s)

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{u}} \; & \sum_{k=0}^{K-1} g(x_k - x_s, u_k - u_s) \\ \text{subject to } & x_{k+1} = f(x_k, u_k) \\ & x_0 = x, \; x_K = x_s \\ & x_k \in \mathcal{X}_k \\ & u_k \in \mathcal{U}_k \end{aligned}$$

This two-stage approach is simpler to analyze but suffers from suboptimal cost of the system trajectory.

Economic MPC uses instead the loss function ℓ directly in the dynamic optimization problem

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \sum_{k=0}^{K-1} \ell(x_k, u_k) \\ \text{subject to } x_{k+1} &= f(x_k, u_k) \\ x_0 &= x,\ x_K = x_s \\ x_k &\in \mathcal{X}_k \\ u_k &\in \mathcal{U}_k \end{aligned}$$

eliminating the cascaded optimization by replacing it with a single time-scale problem for which the optimization can compute an efficient trajectory, at the cost of a more computationally difficult problem to solve in real-time, since often times the problem is not convex.

Economic MPC is computationally more difficult compared to standard MPC where g is zero at the equilibrium point because there are operating points (x, u) which are not steady states but are economically more convenient:

$$\ell(x, u) < \ell(x_s, u_s).$$

To illustrate the situation, consider the furnace example above, starting from an operation point where the furnace is hot and producing molten iron. The controller will stop the input of coke, thereby reducing the costs but the furnace will keep producing iron while its temperature remains above the melting point of ore, if the MPC horizon K is too short. In other words, we want the furnace to be economical while at the same time being in a operating state for the future.

The economic MPC is naturally an infinite horizon problem because we request long-term profits to be maximised. Moreover an optimal trajectory may not drive the system to a single point equilibrium but rather follow a semi-periodic trajectory (see for instance [2, page 158]). The boundedness is guaranteed by the constraint over the infinite trajectory because by construction, the controller produces an acceptable future trajectory. Such infinite horizon problem problems are computationally very hard.

Solving the problem with a finite horizon is more tractable but because the system is not necessarily driven towards the equilibrium, we cannot assume that the tail, after the horizon K, is small as we did with tracking

MPC. Instead low cost unstable trajectories may emerge (cheap trajectory now, expensive to fix in the future). This is the reason why eMPC are solved by constraining the terminal state $x_K = x_s$.

Imposing $x_K = x_s$ induces a final stage cost $g_K(x_s)$ but ensures that the system is still operational at the end of the horizon and not in an unrecoverable state (unfeasible? high cost?). The increased terminal cost $g_K(x_K)$ affects minimally the closed-loop solution (but if it affects minimally, why adding it?).

3.0.1 Stability and Performance of Economic MPC

In tracking MPC, the terminal constraint $x_K = 0$ was essential in proving that eq. (2.5) was a Lyapunov function for the plant. The key idea was that a valid new trajectory would keep the state close to equilibrium at the tail. However for economic MPC, the same manipulation does not work, since

$$W(x_{i+1}) \leq W(x_i) + \ell(x_s, u_s) - \ell(x_i, u_0^\star(x_i)) \tag{3.1}$$

is not in general smaller than $W(x_i)$: to keep the system at steady-state, the cost may exceed the cost of the state after applying control input $u_0^\star(x)$. In contrast with tracking MPC where the stage cost g_k had the property of being zero at the equilibrium point, in economic MPC, ℓ is not minimized at the steady state and stronger conditions are required to guarantee asymptotic stability. This is outside the scope of this introduction.

Since eMPC can generate semi-periodic trajectories, are these trajectories better than tracking a point? The following theorem guarantees that trajectories created started from an arbitrary initial point are more efficient that tracking a steady-state provided the number of time steps N is sufficiently large:

Theorem 3. *Let $\{x_k,\ k = 0, \ldots, N-1\}$ be a closed-loop trajectory generated by the economic MPC controller with terminal constraint x_s. Then, for any initial condition x_0*

$$\limsup_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \ell(x_k, u_k) \leq \ell(x_s, u_s). \tag{3.2}$$

Proof. The proof relies on the identity eq. (3.1), rewritten as

$$W(x_{i+1}) - W(x_i) \leq \ell(x_s, u_s) - \ell(x_i, u_0^\star(x_i))$$

Summing the first N terms of the series and dividing by N gives

$$\frac{1}{N}\sum_{i=0}^{N-1}\left[W(x_{i+1}) - W(x_i)\right] = \underbrace{\frac{W(x_N) - W(x_0)}{N}}_{\to 0 \text{ when } N \to 0} \leq \frac{1}{N}\sum_{i=0}^{N-1}\left[\ell(x_s, u_s) - \ell(x_i, u_0^\star(x_i))\right]$$

since the quantity $W(x_N) - W(x_0)$ is bounded. After rearranging the terms, we obtain the expression eq. (3.2). $\square$

Chapter 4

Robust MPC

MPC relies on a model of the system. However the model may not capture the system's dynamics for different reasons, *e.g.*

- the true dynamics $x_{k+1} = f(x_k, u_k, w_k)$ contains exogenous (outside of the system and unknown) *unknown* disturbances w_k;
- model mismatch: the system dynamics $x_{k+1} = f(x_k, u_k)$ is an approximation of the real system $x_{k+1} = \tilde{f}(x_k, u_k)$;
- missing dynamics f_z: the system is deterministic but the true Markovian state $[x, z]^\top$ and dynamics
$$\begin{bmatrix} x_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} f(x_k, u_k, z_k) \\ f_z(x_k, u_k, z_k) \end{bmatrix}$$
are unknown;
- the system is a linear approximation $x_{k+1} = \tilde{A}x_k + \tilde{B}u_k$ of a non-linear system;
- time-discretization, quantization, time-varying parameters...

We have seen how feedback takes care to some extent of the disturbances. If prior information on the disturbances is available, *e.g.*

- a finite set of discrete disturbances $w_k \in \{w^0, w^1, \dots, w^p\}$;
- the convex hull (the linear combination) of a finite disturbance set, defining a polytope $w_k \in \text{co}\{w^0, w^1, \dots, w^p\}$;

- a probability distribution $w_k \approx \mathcal{W}$. It is rare to have a true probability distribution; rather one has knowledge of the past distribution.

can it be used in the computation of the optimal control? The scope of this lecture is to obtain a control input sequence that is *robust* against disturbances. We will restrict ourselves to exogenous systems

$$x_{k+1} = f(x_k, u_k, w_k)$$

and assume a finite set of disturbances.

To set up the optimization, cost function and constraints are to be considered. According to the problem to solve, the cost can be

- the nominal cost, the cost when the disturbance is zero;
- worst case cost;
- expected cost, the cost weighted by the probability distribution;
- cost for typical noise? (*e.g.* 95% of cases).

The second aspect is the constraints:

- guaranteed satisfaction of constraints: no matter what happens, satisfy the constraints (*e.g.* a machine breaks outside constraints)
- constraint satisfaction with high probability (*e.g.* it is OK if the temperature in the building is outside the optimal range 3% of the time) and typically evaluated by doing a Monte Carlo simulation;
- bounds on constrained violation (*e.g.* how much the system is allowed to violate the constraints to minimize the risk).

4.1 Linear Systems with Additive Disturbance

We consider in the following linear systems[1] with additive exogenous disturbances and worst case satisfaction of constraints

$$\begin{aligned}
&\min_{\mathbf{u}} \left(\max_{\mathbf{w}} \sum_{k=0}^{K-1} g_k(x_k, u_k) + g_K(x_K) \right) \\
\text{subject to } & x_{k+1} = Ax_k + Bu_k + Ew_k \\
& x_0 = x \\
& x_k \in \mathcal{X}_k \quad \forall w \in \mathbb{W} \\
& u_k \in \mathcal{U}_k
\end{aligned} \tag{4.1}$$

Here the minimization is performed on the worst possible set of disturbances as determined by $\max_{\mathbf{w}}$, while satisfying the constraints. In other words, we allow the disturbance to pick a value w_k knowing the action u_k in an adversary way.

Standard MPC may not suffice to solve such problems. Consider the following example:

$$\begin{aligned}
&\min_{\mathbf{u}} \max_{\mathbf{w}} \sum_{k=1}^{4} x_k^2 + u_k^2 \\
\text{subject to } & x_{k+1} = x_k + u_k + w_k \\
& |x_k| \leq 1, \quad |w_k| \leq 0.5
\end{aligned} \tag{4.2}$$

The condition $|x_5| \leq 1$ implies

$$-1 \leq x_0 + \sum_{k=1}^{4} u_k + \sum_{k=1}^{4} w_k \leq +1.$$

When the disturbances are either $w_k = +0.5$ or $w_k = -0.5$ for all k:

$$\begin{aligned}
-3 \leq x_0 + \sum_{k=1}^{4} u_k \leq -1 \text{ when } w_k = +0.5 \\
+1 \leq x_0 + \sum_{k=1}^{4} u_k \leq +3 \text{ when } w_k = -0.5.
\end{aligned}$$

[1]It looks like that most of the discussion is also valid for a generic system dynamics $x_{k+1} = f_k(x_k, u_k, w_k)$.

no input $\mathbf{u}$ satisfies both inequalities. On the other hand, by choosing the closed loop control input $u_k = -x_k$, the dynamics becomes $x_{k+1} = w_k$ and the problem is feasible, because $|x_k| \le 1,\ \forall k$ irrespectively of the initial condition $|x_0| \le 1$, although possibly not optimal. Standard MPC fails by trying to solve this problem in open loop: it searches a conservative control input $\mathbf{u}$ that works for *all* disturbances and feedback is only added at a later stage by using the measured state as the initial state for the new optimization.

4.2 Robust Control Requires Closed Loop Optimization

True closed loop control[2] such as that implemented by LQR, requires finding a policy $u_k = \pi_k(x_k)$ that is a function of the state x_k. Standard MPC is a proxy to finding the optimal control policy because it solves the optimization problems on the numbers $\mathbf{u}$.[3]

Therefore we look for an input policy for the closed loop optimal control

$$u_k = \pi_k(x_k) \quad k = 0, 1, \ldots, K-1$$

or equivalently a policy given the past disturbances

$$u_k = \theta_k(w_0, \ldots, w_{k-1}) \quad k = 0, 1, \ldots, K-1$$

because the state is Markovian and by using the past disturbances, it is possible to reconstruct state x_k.

4.2.1 Soft-Constrained LQR

Finding a robust optimal closed-loop control policy is often hard; the problem however admits a closed form solution when the state update is linear and the cost quadratic. The soft-constrained LQR has the same cost as LQR

$$V(x) = \min_{\mathbf{u}} \left[\max_{\mathbf{w}} \sum_{k=0}^{K-1} x_k^\top Q x_k + u_k^\top R u_k - \gamma^2 w_k^\top w_k + x_K^\top S x_K \right], \quad Q, S \succeq 0, R \succ 0$$

[2]There is a bit of overloading of the term "closed loop" when it comes to MPC: there is closed loop in the sense that the initial state comes from the measurements and true closed loop when the policy is a function of the state $u_k = \pi_k(x_k)$.

[3]This sentence needs some love.

except for the additional negative cost $-\gamma^2 w_k^\top w_k$. This term works as a soft bound on the energy of the disturbance w_k and is irrelevant in the minimization with respect to $\mathbf{u}$ thereby not changing the optimal control input sequence. If γ is sufficiently large, $\max_{\mathbf{w}}$ makes the problem concave and therefore solvable.

The solution of this dynamic programming problem consists in assuming that $V_k(x) = x^\top P_k x$ is quadratic in x, as it was the case for LQR. The resulting Isaac equation is solved by dynamic programming by induction in the same way as it was done for Bellman's equation

$$V_k(x) = \min_u \max_w x^\top Q x + u^\top R u - \gamma^2 w^\top w + V_{k+1}(Ax + Bu + Dw)$$

and gives the optimal disturbance $w_k^\star(x)$ and the optimal input $u_k^\star(x)$. For the optimal disturbance, the maximization of the inner problem gives

$$\hat{w}_k(x,u) = -(D^\top P_{k+1} D - \gamma^2 I)^{-1} D^\top P_{k+1}(Ax + Bu) \doteq \Lambda x + \Gamma u$$

where $\hat{w}_k(x,u)$ is a maximum only if the Hessian of the cost to go is negative, that is when

$$\gamma^2 I \succ D^\top P_{k+1} D.$$

To compute the optimal control input, we assume that $u_k^\star(x) = Kx$; by plugging it into

$$w_k^\star(x,u) = \Lambda x + \Gamma u = \underbrace{(\Lambda + \Gamma H)}_{H} x$$

the cost-to-go has indeed the form $V_k(x) = x^\top P_k x$ with

$$P_k = Q + K^\top R K - \gamma^2 H^\top H + (A + BK + DH)^\top P_{k+1}(A + BK + DH)$$

from which we derive

$$K_k =$$

Similarly to the LQR case, the entire computation can be performed offline; the online part in a receding horizon scheme is $u_0^\star(x) = K_0 x$.

Note that in the absence of disturbances, open and closed loop MPC give the same trajectory. In closed loop different disturbances will give different optimal control inputs whereas in open loop, standard MPC will compute one optimal sequence $\mathbf{u}^\star$ for all disturbances.

4.2.2 Feedback MPC

The soft-constrained LQR approach does not admit constraints. Feedback MPC reintroduces them while allowing one to optimize over policies $\pi_k(x)$ for true closed-loop feedback. The optimization over generic function $\pi_k(x)$ is difficult: the problem is rendered tractable by parametrizing the feedback control law $u_k(x) = \pi_k(x_k, \mathbf{v})$ with the parameters $\mathbf{v}$ and to optimize over them

$$\begin{aligned} \min_{\mathbf{v},\mathbf{x}} \max_{\mathbf{w}} & \sum_{k=0}^{K-1} g_k(x_k, \pi_k(x_k, \mathbf{v})) + g_K(x_K) \\ \text{subject to } & x_{k+1} = f(x_k, \pi_k(x_k, \mathbf{v}), w_k) \\ & x_0 = \mathrm{x} \\ & x_k \in \mathcal{X}_k \quad \forall \mathbf{w} \in \mathbb{W} \\ & \pi(x_k, \mathbf{v}) \in \mathcal{U}_k \end{aligned} \tag{4.3}$$

The choice of the parametrization can generate a spectrum of policies: for example choosing $u_k(x_k) = \pi_k(x_k, \mathbf{v}) = v_k$ with $\mathbf{v} \in \mathbb{R}^{K-1}$ amounts to an open loop policy. A closed loop is any function $u_k(x_k) = \pi_k(x_k)$ with $\mathbf{v} \in \mathbb{R}^{\infty}$: as seen earlier, this can be done only in special cases such as LQR. Between these two extremes, a common approach is to restrict the policies to the (linear) combination of a set of functions $\theta_m(x)$

$$u_k(x_k) = \pi_k(x_k, \mathbf{v}) = \sum_{m=1}^{M} v_{km} \theta_m(x_k), \quad \mathbf{v} \in \mathbb{R}^{KM}.$$

We consider here the affine control law $\pi_k(x_k, \mathbf{v}) = v_k + Lx_k$ where L is given. The feedback system dynamics is determined by

$$x_{k+1} = Ax_k + B \underbrace{(v_k + Lx_k)}_{\pi_k(x_k, \mathbf{v})} + Ew_k = (A + BL)x_k + Bv_k + Ew_k$$

where v_k now appears as the effective control input. With this affine policy, the sequence $\mathbf{v}^\star$ is the open-loop optimal policy while the feedback gain L rejects the disturbances $\mathbf{w}$. In the case of example 4.2 the choice $L = -1$ yields the control input $u_k = -x_k + v_k$ and changes the system dynamics to $x_{k+1} = w_k$. It is however not critical to select L because there is still the freedom in choosing the control input v_k.

Note that in the absence of disturbances, there is no advantage in this parametrization because v_k can include the term Lx_k computed for the nominal system trajectory.

4.2.3 Joint Optimization of L and v

An extension of the previous case is to optimize also L_k. The naive optimization using $u_k = L_k x_k + v_k$ renders the optimization non-convex. The problem remains convex if the control policies are expressed as a function of the *past* disturbances $w_i,\ i < k$

$$u_k = \sum_{i=0}^{k-1} M_{ki} w_i + v_k.$$

and the optimization is done on $\mathbf{M}$

$$\begin{aligned} \min_{\mathbf{x},\mathbf{M},\mathbf{x}} \max_{\mathbf{w}} & \sum g(x_k, u_k) + g(x_K) \\ \text{subject to } & x_{k+1} = f(x_k, u_k, w_k) \\ & x_0 = \mathrm{x} \\ & x_k \in \mathcal{X}_k \quad \forall \mathbf{w} \in \mathbb{W} \\ & u_k \in \mathcal{U}_k \quad \forall \mathbf{w} \in \mathbb{W} \\ & u_k = \sum_{i=0}^{k-1} M_{ki} w_i + v_k. \end{aligned}$$

since the knowledge of the past disturbances is sufficient to reconstruct the current state x_k

$$x_{k+1} = f(x_k, u_k, w_k).$$

4.2.4 Computational Complexity

The problem eq. (4.3) is a convex problem. It has computational complexity $n \times K \times W$, where n is the dimension of the state, K the time-horizon and W the number of possible disturbances. If $\mathbb{W}$ is a polytope, W is the number of vertices and is exponential in the dimensions of w or, if w is a random disturbance, W could be the points from a Monte Carlo sampling.

Chapter 5

System Identification

LQR and MPC require a state space model of the system and the measurement function

$$x_{t+1} = f(x_t, u_t), \quad y_t = h(x_t, u_t).$$

Obtaining a model requires determining

- the Markovian state x, so that x_{t+1} is determined by x_t and u_t;
- the dynamics $f(x, u)$ of the system;
- the measurement function $h(x, u)$.

In the rest of this chapter we only consider the case of a noiseless linear time-invariant update and observation:

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t. \end{aligned} \tag{5.1}$$

State space representations are not unique: under the invertible transformation T such that $x = Tw$, the system eq. (5.1) becomes[1]

$$\begin{aligned} w_{t+1} &= T^{-1}ATw_t + T^{-1}Bu_t \\ y_t &= CTw_t + Du_t \end{aligned} .$$

Nevertheless the input-output behavior, the relation between the input u and the output y, remains the same. This can be seen by checking that their

[1] I introduce D which is not in the lecture but is in the tutorials.

response is the same for the same input, *e.g.* an input response applied to the k input

$$u_{k,0} = e_k \delta_t, \quad \delta_t = \begin{cases} 1 & \text{for } t = 0 \\ 0 & \text{otherwise} \end{cases}.$$

Assuming a system at rest, we have for the state x the dynamics

$$x_0 = 0,\ x_1 = Be_k,\ x_2 = ABe_k,\ x_3 = A^2Be_k,\ \ldots$$

whereas for w

$$w_0 = 0,\ w_1 = T^{-1}Be_k,\ w_2 = T^{-1}ABe_k,\ w_3 = T^{-1}A^2Be_k,\ \ldots$$

and for both systems, the output y_t is the same,

$$y_0 = De_k,\ y_1 = CBe_k,\ y_2 = CABe_k,\ y_3 = CA^2Be_k,\ \ldots$$

For a general system with m inputs and p outputs, the sequence of matrices in $R^{p\times m}$ describing the output of the system dynamics

$$\{D,\ CB,\ CAB,\ CA^2B,\ \ldots\} \tag{5.2}$$

is called the *Markov parameters.* The matrix element $\left[CA^tB\right]_{ij}$ represent the output i to an unit impulse on the input j after a time $t > 0$.

5.0.1 Controllability and Observability

Consider the problem of driving a system from $x_0 = 0$ to a desired $\bar{x} \in \mathbb{R}^n$. In one step, input u_0 will take the system to

$$x_1 = Bu_0$$

that is, to any state in the image of B. In two steps, the control sequence $\{u_0, u_1\}$ will drive the system to

$$x_2 = Bu_1 + ABu_0$$

and therefore to any state in the image of $\begin{bmatrix} B & AB \end{bmatrix}$. In n steps, the system can be driven to any state in the image of the *controllability matrix*

$$\mathcal{C} \doteq \begin{bmatrix} B & AB & A^2B & \ldots & A^{n-1}B \end{bmatrix}. \tag{5.3}$$

$\mathcal{C}$'s column space is also the largest space that can be reached by the system: the matrix A^n for an additional step is linearly dependent on $I, A, \ldots, A^{n-1}$

by the Cayley-Hamilton theorem. In general $\mathcal{C}$'s column space does not span the whole space $\mathbb{R}^n$: this is the case only when $\mathcal{C}$'s rank is n.

For a full rank controllability matrix, the desired state $\bar{x}$ can be reached in at most n steps since

$$\bar{x} = \mathcal{C} \begin{bmatrix} u_{t-1} \\ u_{t-2} \\ \vdots \\ u_0 \end{bmatrix} = \mathcal{C}\mathbf{u}$$

by using the control input $\mathbf{u} = \mathcal{C}^{-1}\bar{x}$ when the input is scalar or by using the pseudoinverse $\mathbf{u} = \mathcal{C}^{\dagger}\bar{x}$ in the case of a MIMO system, where $\mathcal{C}$ is a fat matrix.

Similarly, consider the problem of detecting the effect of an initial condition x_0 on the output of a system with no control input. At $t = 0$

$$y_0 = Cx_0$$

therefore any x_0 in C's kernel is non-observable. At $t = 1$, the output is

$$y_1 = CAx_0$$

and any x_0 in the kernel of $\begin{bmatrix} C \\ CA \end{bmatrix}$ is non-observable. In n steps, any initial state in the kernel of the *observability matrix*

$$\mathcal{O} \doteq \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{5.4}$$

is non-observable. As we have seen for the controllability matrix, an additional step, CA^n does not increase $\mathcal{C}$'s row rank because by the Cayley-Hamilton theorem, A^n is a linear combination of $\{I, A, \dots, A^{n-1}\}$.

We can use the observability matrix to find the initial state x_0 given the output sequence $\mathbf{y} = \begin{bmatrix} y_0 & y_1 & \dots & y_{n-1} \end{bmatrix}^\top$, since

$$\mathbf{y} = \mathcal{O}x_0.$$

Assuming $\mathcal{O}$ has rank full, if the system has only one output (*i.e.* the matrix C has only row), $\mathcal{O}$ is square and invertible and the initial state is found by $x_0 = \mathcal{O}^{-1}\mathbf{y}$; otherwise, in case of a multiple output system, $\mathcal{C}$ is thin and we need to use $\mathcal{O}$'s pseudoinverse, $x_0 = \mathcal{O}^{\dagger}\mathbf{y}$.

5.0.2 System Identification from Data

We want to derive a state-space representation exclusively from input-output data. We restrict our attention to systems that are both controllable and observable, so that the relation between states and input/output signals is well-posed (?). This also excludes the case of states that have no effect on I/O (is this not the same as the statement before?) We also want the minimal realization, the minimum number of states relevant for the system.

The Ho-Kalman algorithm allows one to derive a minimal realization of the system in the form of the matrices A, B, C from impulse response data. The algorithm consists of the following steps:

1. construct the Hankel matrix $\mathcal{H}$ from the Markov parameters;
2. factorize $\mathcal{H}$ into the matrices $\mathcal{O}$ and $\mathcal{C}$;
3. derive the realization of A, B, C.

Step 1: Construct $\mathcal{H}$

The impulse responses of the system are collected as Markov parameters eq. (5.2). The Hankel matrix is a banded matrix

$$\mathcal{H} \doteq \mathcal{O}\mathcal{C} = \begin{bmatrix} CB & CAB & CA^2B & \dots & CA^{n-1}B \\ CAB & CA^2B & \dots & & CA^nB \\ CA^2B & \ddots & & & \dots \\ \vdots & & & & \vdots \\ CA^{n-1}B & \dots & & & CA^{2n-2}B \end{bmatrix} \tag{5.5}$$

with the Markov parameters along the diagonals. Only for SISO systems is each entry a scalar, for MIMO systems each entry has size $p \times m$.

We are seeking a minimal representation of a controllable and observable system. Since both $\mathcal{O}$ and $\mathcal{C}$ are full rank, so must $\mathcal{H}$. The rank n is however unknown: the approach is to construct an extended Hankel matrix $\mathcal{H}_{k,\ell} \in \mathbb{R}^{k\times\ell}$ and choose k and ℓ larger than the expected value n. In doing so, $\mathcal{H}_{k,\ell}$ will be singular. To determine n we select the largest upper left square minor of $\mathcal{H}_{k,\ell}$ that is non-singular.

This is better explained by an example: consider a time-discrete frictionless cart

$$\begin{aligned} v_{t+1} &= v_t + u_t \\ z_{t+1} &= z_t + v_t \longrightarrow x_t = \begin{bmatrix} v_t \\ z_t \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 \end{bmatrix} \\ y_t &= z_t \end{aligned}$$

subjected to an impulse response $u_t = \delta_t$. The time evolution is

$$\begin{aligned} v_0 &= 0, \quad z_0 = y_0 = 0 \\ v_1 &= 1, \quad z_1 = y_1 = 0 \\ v_2 &= 1, \quad z_2 = y_2 = 1 \\ v_3 &= 1, \quad z_3 = y_3 = 2 \\ &\dots \qquad\quad \dots \end{aligned}$$

Let us arbitrarily pick $k = 4$ and $\ell = 6$. The corresponding extended Hankel matrix is the matrix whose rows are the system output, each row shifted to the left:

$$\mathcal{H}_{4,6} = \begin{bmatrix} y_0 & y_1 & y_2 & y_3 & y_4 & y_5 \\ y_1 & y_2 & y_3 & \dots & & \\ y_2 & y_3 & \dots & & & \\ y_3 & \dots & & & & y_8 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}.$$

The 1×1 top left submatrix

$$\begin{bmatrix} 0 \end{bmatrix}$$

has rank zero, the 2×2 top left submatrix

$$\begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$$

has rank 2 and so have the other square top left submatrices: the rank does not increase if more rows or columns are added. A maximum rank of 2 is to be expected because the system is 2-dimensional.

The computation of the rank is however not sufficiently robust against perturbation of the coefficients: the next section presents a better approach.

Step 2: Factorize $\mathcal{H}$

Factorize $\mathcal{H}_{k,\ell}$ as the product of two matrices $\mathcal{O}_k$ and $\mathcal{C}_\ell$

$$\mathcal{O}_k = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{k-1} \end{bmatrix}, \qquad \mathcal{C}_\ell = \begin{bmatrix} B & AB & \dots & A^{\ell-1}B \end{bmatrix}. \tag{5.6}$$

The factorization is not unique and different factorizations give different space state representations of the system, all sharing the same input-output response. Indeed consider a change of basis $\tilde{A} = T^{-1}AT$, $\tilde{B} = T^{-1}B$ and $\tilde{C} = CT$. The observability matrix changes as

$$\tilde{\mathcal{O}} = \begin{bmatrix} \tilde{C} \\ \tilde{C}\tilde{A} \\ \vdots \end{bmatrix} = \begin{bmatrix} CT \\ CTT^{-1}AT \\ \vdots \end{bmatrix} = \mathcal{O}T$$

whereas the controllability matrix changes as $\tilde{\mathcal{C}} = T^{-1}\mathcal{C}$ but the product $\tilde{\mathcal{O}}\tilde{\mathcal{C}} = \mathcal{O}\mathcal{C}$ remains the same.

SVD is a computationally fast and robust method to obtain the decomposition of the matrix $\mathcal{H}_{k,\ell} = U\Sigma V^\top$

$$\mathcal{H}_{k,\ell} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0_{n\times(\ell-n)} \\ 0_{(k-n)\times n} & 0_{(k-n)\times(\ell-n)} \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^\top = U_1\Sigma_1 V_1^\top.$$

where the dimension n of the diagonal matrix Σ_1 is the rank of $\mathcal{H}_{k,\ell}$ and the size of the state space. The remaining elements of Σ are zero, because increasing k and ℓ does not increase $\mathcal{H}_{k,\ell}$ rank.

A balanced factorization is

$$\mathcal{H}_{k,\ell} = \underbrace{U_1\sqrt{\Sigma_1}}_{\mathcal{O}_k}\underbrace{\sqrt{\Sigma_1}V_1^\top}_{\mathcal{C}_\ell}.$$

For the frictionless cart

$$\Sigma = \begin{bmatrix} 21.90 \\ 2.092 \\ 0 \\ 0 \end{bmatrix}$$

and the decomposition gives

$$\mathcal{O}_k = \begin{bmatrix} -1.547 & -1.112 \\ -2.033 & -0.4826 \\ -2.519 & 0.1467 \\ -3.005 & 0.7759 \end{bmatrix}$$

$$\mathcal{C}_\ell = \begin{bmatrix} -0.7345 & -1.150 & -1.566 & -1.982 & -2.397 & -2.813 \\ 1.022 & 0.7010 & 0.3800 & 0.0589 & -0.2621 & -0.583 \end{bmatrix}.$$

Step 3: Obtain the Realization A, B, C

C and B are the first entries in $\mathcal{O}_k$ and $\mathcal{C}_\ell$, eq. (5.6). A can be obtained from the shifted Hankel matrix, obtained discarding the first column of $\mathcal{H}_{k,\ell}$ and adding the subsequent Markov parameters as the last column (or alternatively, discarding the first row and adding the subsequent Markov parameters as the last row)

$$\mathcal{H}^{\uparrow}_{k,\ell} \doteq \begin{bmatrix} CAB & CA^2B & CA^3B & \dots & CA^{\ell}B \\ CA^2B & CA^2B & \dots & & CA^{\ell+1}B \\ CA^3B & \ddots & & & \dots \\ \vdots & & & & \\ CA^kB & \dots & & & \dots \end{bmatrix} = \mathcal{O}_k A \mathcal{C}_\ell.$$

Then A becomes

$$A = \mathcal{O}_k^{\dagger} \mathcal{H}^{\uparrow}_{k,\ell} \mathcal{C}_\ell^{\dagger}.$$

For the frictionless cart, we obtain

$$A = \begin{bmatrix} 1.202 & -0.2616 \\ 0.156 & 0.7980 \end{bmatrix}, \quad B = \begin{bmatrix} -0.7345 \\ 1.0220 \end{bmatrix}, \quad C = \begin{bmatrix} -1.5470 & -1.1118 \end{bmatrix}$$

The Ho-Kalman algorithm gives a different representation of the matrices A, B and C[2] but the same observed dynamics. (How can I see this?)

Note also that the frictionless cart is only marginally stable as A's eigenvalues are both 1. The approach can be also for unstable systems provided the experiment is stopped before the system becomes irrecoverable. Otherwise, system identification on unstable systems must be performed with feedback stabilization, which is outside this introduction.

[2]There is a coordinate transformation T which I cannot find: A is defective: should I be using the Jordan form?

Chapter 6

Data-Enabled Predictive Control

This lecture presents a data-based representation of a linear system. This description generates valid inputs and outputs without using a state-based representation and avoids system identification based *e.g.* on the Ho-Kalman algorithm.

6.1 State-Based System Response

The time evolution of the linear system

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t \end{aligned}$$

from the initial state x_0 and under the control input $\mathbf{u}$ is

$$y_t = \underbrace{CA^t x_0}_{\text{free response}} + \underbrace{\sum_{j=0}^{t-1} CA^{t-1-j} Bu_j +}_{\text{convolution}} \underbrace{Du_t}_{\text{feed through}} \tag{6.1}$$

or, in matrix form,

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{L-1} \end{bmatrix} = \underbrace{\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{L-1} \end{bmatrix}}_{\doteq \mathcal{O}_L} x_0 + \underbrace{\begin{bmatrix} D & 0 & 0 & \dots & 0 \\ CB & D & 0 & \dots & 0 \\ CAB & CB & D & & \vdots \\ \vdots & & & & \\ CA^{L-2}B & CA^{L-3}B & \dots & CB & D \end{bmatrix}}_{\doteq \mathcal{G}_L} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_{L-1} \end{bmatrix} \tag{6.2}$$

$\mathcal{O}_L$ is the observability matrix and $\mathcal{G}_L$ the Markov coefficients describing the input-output of the system when the initial state is $x_0 = 0$.

For the rest of this chapter, we will consider for simplicity SISO systems, $u \in \mathbb{R}$ and $y \in \mathbb{R}$, that are both controllable and observable. The additional complication for MIMO arises from the need to keep track of m inputs and p outputs.

To keep the notation compact[1], we will indicate a vector of length L by

$$\mathbf{w}_L \doteq \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{L-1} \end{bmatrix}.$$

Using this notation, the expression eq. (6.2) is rewritten as

$$\mathbf{y}_L = \mathcal{O}_L x_0 + \mathcal{G}_L \mathbf{u}_L$$

where (for SISO) $\mathcal{O}_L \in \mathbb{R}^{L \times n}$ and $\mathcal{G}_L \in \mathbb{R}^{L \times L}$ is a square matrix.

We have seen in the previous chapter that the rank of $\mathcal{O}_L$ can at most be n: if $L = n$, $\mathcal{O}_L$ is invertible since the system is assumed to be observable, and the initial state x_0 is uniquely determined by

$$x_0 = \mathcal{O}_L^{-1}(\mathbf{y}_L - \mathcal{G}_L \mathbf{u}_L)$$

if the control input $\mathbf{u}_L$ is known. If $L > n$ the system of equations is overdetermined, the initial state x_0 can be reconstructed only if $\mathbf{y}_L$ is a valid output sequence, that is, if the vector $\mathbf{y}_L - \mathcal{G}_L \mathbf{u}_L$ is in the column image of $\mathcal{O}_L$.

[1]Should we have this definition somewhere earlier?

In the different representation of behavioral system theory, the terms are rearranged as

$$\begin{bmatrix} \mathbf{u}_L \\ \mathbf{y}_L \end{bmatrix} = \underbrace{\begin{bmatrix} I_L & 0 \\ \mathcal{G}_L & \mathcal{O}_L \end{bmatrix}}_{\Lambda_L} \begin{bmatrix} \mathbf{u}_L \\ x_0 \end{bmatrix}$$

where $\Lambda_L \in \mathbb{R}^{2L\times(L+n)}$ and has rank $L+n$ when $L \geq n$, since one is free to choose the initial state $x_0 \in \mathbb{R}^n$ and L control inputs.

This formulation highlights the fact that trajectories of inputs $\mathbf{u}_L$ and outputs $\mathbf{y}_L$ must be in the column image of Λ_L. The size of the matrix Λ_L grows as the length of the time sequence one tries to verify.

In this approach inputs and outputs are treated on equal footing. This is convenient for systems where is no such clear-cut distinction, for instance a complex electric circuit, where voltages and currents are mutually dependent.

The matrix Λ_L is a full description of the system equivalent to the one generated by the state space system matrices A, B, C, D and can predict the future K outputs, $\mathbf{y}_K^{(\text{future})} = \{y_n, \dots, y_{n+K-1}\}$, given the future K inputs, $\mathbf{u}_K^{(\text{future})} = \{u_n, \dots, u_{n+K-1}\}$,

$$\begin{bmatrix} \mathbf{u}_n^{(\text{past})} \\ \mathbf{u}_K^{(\text{future})} \\ \mathbf{y}_n^{(\text{past})} \\ \mathbf{y}_K^{(\text{future})} \end{bmatrix} = \underbrace{\begin{bmatrix} I_{n+K} & 0 \\ \mathcal{G}_{n+K} & \mathcal{O}_{n+K} \end{bmatrix}}_{\Lambda_L} \begin{bmatrix} \mathbf{u}_n^{(\text{past})} \\ \mathbf{u}_K^{(\text{future})} \\ x_0 \end{bmatrix}$$

with the matrix Λ_L appropriately extended. This can be done, since Λ_L has been derived from the state space system matrices, and the initial state x_0 is either known or can be uniquely derived from the past inputs $\mathbf{u}_n^{(\text{past})} = \{u_0, \dots, u_{n-1}\}$ and outputs $\mathbf{y}_n^{(\text{past})} = \{y_0, \dots, y_{n-1}\}$.

6.2 Behavioral Model Approach

There is an alternative approach to the prediction of the future K outputs and the behavior theory formulation makes it convenient to express. The main observation is that input and outputs $\begin{bmatrix} \mathbf{u}_L & \mathbf{y}_L \end{bmatrix}^\top$ can only live in a subspace of dimension $2n + K = n + L$ in the space $\mathbb{R}^{2L}$: this subspace is spanned for instance by $n + L$ independent *experiments*

$$\begin{bmatrix} \mathbf{u}_L^{(1)} \\ \mathbf{y}_L^{(1)} \end{bmatrix}, \begin{bmatrix} \mathbf{u}_L^{(2)} \\ \mathbf{y}_L^{(2)} \end{bmatrix}, \begin{bmatrix} \mathbf{u}_L^{(3)} \\ \mathbf{y}_L^{(3)} \end{bmatrix}, \dots, \begin{bmatrix} \mathbf{u}_L^{(n+L)} \\ \mathbf{y}_L^{(n+L)} \end{bmatrix}.$$

Therefore all and the only valid input-output sequences are of the form

$$\begin{bmatrix}\mathbf{u}_L\\ \mathbf{y}_L\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{u}_L^{(1)} & \mathbf{u}_L^{(2)} & \mathbf{u}_L^{(3)} & \dots & \mathbf{u}_L^{(n+L)}\\ \mathbf{y}_L^{(1)} & \mathbf{y}_L^{(2)} & \mathbf{y}_L^{(3)} & \dots & \mathbf{y}_L^{(n+L)}\end{bmatrix}}_{H} g$$

with $g \in \mathbb{R}^{n+L}$. Note that this is effectively a change of coordinates: g is in general not more the vector of control inputs and initial state anymore[2].

To construct H while being able to predict K steps in the future, at least $n + L = 2n + K$ trajectories are needed, each of length $L = n + K$. An efficient way to produce the required data is to acquire a *single* experiment of length $2L+n-1$ and to generate $n+L$ synthetic experiments by rearranging the control input sequence $\{u_0, u_1, u_2, \dots\}$ as the Hankel matrix

$$\mathcal{H}_L(\mathbf{u}) = \begin{bmatrix} u_0 & u_1 & u_2 & \dots & u_{L+n-1}\\ u_1 & u_2 & u_3 & & \\ u_2 & u_3 & u_4 & & \\ \vdots & & & & \\ u_{L-1} & u_L & u_{L+1} & \dots & u_{2L+n-2}\end{bmatrix}$$

and, equivalently, $\mathcal{H}_L(\mathbf{y})$ from the measured data $\mathbf{y}$. The K step predictor becomes

$$\begin{bmatrix}\mathbf{u}_n^{(\text{past})}\\ \mathbf{u}_K^{(\text{future})}\\ \mathbf{y}_n^{(\text{past})}\\ \mathbf{y}_K^{(\text{future})}\end{bmatrix} = \begin{bmatrix}\mathcal{H}_L(\mathbf{u}_{\text{data}})\\ \mathcal{H}_L(\mathbf{y}_{\text{data}})\end{bmatrix} g$$

with $g \in \mathbb{R}^{2n+K}$ and $\mathbf{y}^{\text{future}} \in \mathbb{R}^K$ unknowns: the problem is determined because there $2(n + K)$ unknowns and an equal number of equations.

6.3 Model-Based Predictive Control

The behavioral model permits one to express MPC using the matrix H that generates valid future control inputs $\mathbf{u}$ and future outputs $\mathbf{y}$ compatible with

[2]The new matrix H corresponds to Λ_L when the experiments are constructed in the following ways: for the first L experiments, the initial condition is 0 and only one input at the time is activated with unit amplitude: for the first experiment, $u_0 = \delta_{t-1}$ at time $t = 1$, for the second experiment $u_1 = \delta_{t-2}$ at time $t = 2$.... For the remaining n experiments, the inputs are not activated and one element of the basis of the state space is activated at the time.

the past data:

$$\begin{aligned}
\min_{g} \quad & \sum_{k=0}^{K-1} g_k(y_k, u_k) + g_K(y_K) \\
\text{subject to} \quad & \begin{bmatrix} \mathcal{H}_L(\mathbf{u}_{\text{data}}) \\ \mathcal{H}_L(\mathbf{y}_{\text{data}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \\ \mathbf{y} \end{bmatrix} \\
& u_k \in \mathcal{U}_k \ \forall k \\
& y_k \in \mathcal{Y}_k \ \forall k
\end{aligned}$$

The state x of the system is not required by the DeePC approach and therefore it cannot appear in the cost function.

From a computational point of view, this problem is more complex than standard tracking MPC:

- it requires a much larger memory footprint to hold H instead of the system matrices A, B, C, D; and
- it needs to optimize on $g \in \mathbb{R}^{2n+K}$, where K degrees of freedom are the future input $\mathbf{u}$ and the remaining $2n$ terms of g are determined by the initial conditions $\begin{bmatrix} \mathbf{u}_{\text{past}} & \mathbf{y}_{\text{past}} \end{bmatrix}^\top$. The variables $\mathbf{u}$ and $\mathbf{y}$ are fully determined by g via the H matrix: although they are redundant by a mathematical point of view, they are usually explicitly used in the numerical solution of the optimization.

6.4 Persistence of Excitation

To construct H, the experiment data $(\mathbf{u}_{\text{data}}, \mathbf{y}_{\text{data}})$ must represent the dynamics of the system: dynamics that is not excited cannot be controlled.

The identification of unstable systems requires additional care: typically the unstable system is prestabilized and for identification the system is perturbed around this equilibrium point.

Indeed it is not sufficient to simply collect the input/output data from a stabilized system to guarantee the identification because input and output are constrained by the controller. For instance with a proportional controller, the input/output data would be of the form

$$\begin{bmatrix} \mathbf{u}_{\text{data}} \\ K\mathbf{u}_{\text{data}} \end{bmatrix}$$

and therefore the experimental data span the subspace resulting from the intersection of the plant and the controller spaces. To probe the full sysem dynamics it is sufficient to introduce an external input excitation v with enough persistent excitation, such that the input to the plant becomes now $v - Ky$.

The perturbation may be also applied on more inputs simultaneously to keep the system closer to a stable state: in a quadcopter, for instance, activating a propeller at the time would rapidly render the system uncontrollable and the identification is done by activating all or pairs of rotors[3].

6.5 Extention of DeePC to the MIMO Case

The *lag* of a system is defined as the smallest ℓ such that the observability matrix

$$\mathcal{O}_\ell = \begin{bmatrix} C \\ CA \\ \dots \\ CA^{\ell-1} \end{bmatrix}$$

has full rank n. For a SISO system, we saw that $\ell = n$, under the assumption that the system is observable.

For a MIMO system of input size m, output p and state space n, a past trajectory of length ℓ is needed to determine the initial condition and an input/output sequence of length

$$T \geq (m+1)(\ell + K) + n - 1$$

for constructing H.

Since $n \geq \ell$ always, a practical approach to write H is to assume that $\ell = n$, where n is determined by first principles.

6.6 The Effect of Noise

The rank of the H matrix is $n + L$ (or $n + mL$ for a MIMO system). If the data is noiseless, additional columns $\begin{bmatrix} \mathbf{u}_L & \mathbf{y}_L \end{bmatrix}^\top$ do not further increase the rank of H. The presence of noise in the data will spuriously increase ita rank. SVD of H will however reveal this by the small values of the singular eigenvalues past the element $n + L$. The spurious increase of rank

[3]Not clear here: if I activate the sum of two controllers, would I not need to perturb also with the difference?

will manifest also in the controller using a large g and therefore large control inputs to affect the uncontrollable outputs[4].

One can force the H matrix to have rank $n + L$ using the nested optimization

$$\min_{\mathbf{u},\mathbf{y},g,\hat{\mathbf{u}}_{\text{data}},\hat{\mathbf{y}}_{\text{data}}} \sum_{k=0}^{K-1} g_k(y_k, u_k) + g_K(y_K) \tag{6.3}$$

$$\text{subject to} \quad (\hat{\mathbf{u}}_{\text{data}}, \hat{\mathbf{y}}_{\text{data}}) = \arg\min ||\hat{\mathbf{u}}_{\text{data}} - \mathbf{u}_{\text{data}}||^2 + ||\hat{\mathbf{y}}_{\text{data}} - \mathbf{y}_{\text{data}}||^2 \\ \text{subject to rank} \left(\begin{bmatrix} \mathcal{H}_L(\hat{\mathbf{u}}_{\text{data}}) \\ \mathcal{H}_L(\hat{\mathbf{y}}_{\text{data}}) \end{bmatrix} \right) = L + n \tag{6.4}$$

$$\begin{bmatrix} \mathcal{H}_L(\hat{\mathbf{u}}_{\text{data}}) \\ \mathcal{H}_L(\hat{\mathbf{y}}_{\text{data}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{data}} \\ \mathbf{u} \\ \mathbf{y}_{\text{data}} \\ \mathbf{y} \end{bmatrix} \tag{6.5}$$

$$u_k \in \mathcal{U}_k \ \forall k \\ y_k \in \mathcal{Y}_k \ \forall k \tag{6.6}$$

by using the closes points $(\hat{\mathbf{u}}_{\text{data}}, \hat{\mathbf{y}}_{\text{data}})$ to the real data $(\mathbf{u}_{\text{data}}, \mathbf{y}_{\text{data}})$ to construct H. Using a quadratic cost for the nested optimization to remove the noise is arbitrary but a different choice would make little practical difference.

The optimization is computationally hard because the computation of the rank makes the problem non-convex. It can however be simplified by the four following transformations:

- impose the additional constraint that g has at most $L + n$ elements: $||g||_0 \leq L + n$. This is equivalent to the rank and does not therefore alter the original problem;

- drop the rank constraint in eq. (6.4). The arg min problem becomes trivial because it picks $(\hat{\mathbf{u}}_{\text{data}}, \hat{\mathbf{y}}_{\text{data}})$ to be $(\mathbf{u}_{\text{data}}, \mathbf{y}_{\text{data}})$ but it may produce an infeasible problem. The problem still returns a finite number of elements for g because of the 0-norm;

- dispense with the nested optimization altogether, eq. (6.4), and replace the 0-norm by a 1-norm constraint $||g||_1 \leq \alpha$. This is the Lasso constraint to the sparsity approach and makes the problem convex. The value of α determines the sparsity of the solution. This value must

[4]We have assumed that the matrix is observable

be tuned, but this introduces no more variables that the previous one, since the state of the system n is (often) not known.

- remove the 1-norm constraint and add it as a renormalization factor into the objective function eq.(6.3)

  $$\sum_{k=0}^{K-1} g_k(y_k, u_k) + g_K(y_K) + \lambda_g ||g||_1.$$

  A larger λ_g increases robustness to noise but affect optimality and corresponds to selecting a model with smaller n. The posteriori sparsity $||g||_0$ of g gives information about the model size.

The problem so transformed

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y},g} & \sum_{k=0}^{K-1} g_k(y_k, u_k) + g_K(y_K) + \lambda_g ||g||_1 \\ \text{subject to } & \begin{bmatrix} \mathcal{H}_L(\mathbf{u}_{\text{data}}) \\ \mathcal{H}_L(\mathbf{y}_{\text{data}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} \\ \mathbf{y} \end{bmatrix} \\ & u_k \in \mathcal{U}_k \ \forall k \\ & y_k \in \mathcal{Y}_k \ \forall k \end{aligned}$$

makes the problem efficient to solve and usable in practice.

Noise may also be present in the "initial conditions" $(\mathbf{u}_{\text{past}}, \mathbf{y}_{\text{past}})$. This is taken care by an additional regularization term $\lambda_\sigma ||\sigma||_1$

$$\begin{aligned} \min_{\mathbf{u},\mathbf{y},g,\sigma} & \sum_{k=0}^{K-1} g_k(y_k, u_k) + g_K(y_K) + \lambda_g ||g||_1 + \lambda\sigma ||\sigma||_1 \\ \text{subject to } & \begin{bmatrix} \mathcal{H}_L(\mathbf{u}_{\text{data}}) \\ \mathcal{H}_L(\mathbf{y}_{\text{data}}) \end{bmatrix} g = \begin{bmatrix} \mathbf{u}_{\text{past}} \\ \mathbf{u} \\ \mathbf{y}_{\text{past}} + \sigma \\ \mathbf{y} \end{bmatrix} \\ & u_k \in \mathcal{U}_k \ \forall k \\ & y_k \in \mathcal{Y}_k \ \forall k \end{aligned} .$$

6.7 Concluding Remarks

- Data-driven control is a control approach that does not make use of the state of the system: if the state is known and its values need to be controlled, then a model-based approach may work better[5].
- Sometimes prior information (weight, moment of inertia) is available: system identification can typically incorporate this information. On the contrary, DeePC is agnostic to this information.
- DeePC strongly relies on a LTI system. An extension to non-linear problems, while possible, is cumbersome.
- The computational complexity is larger in terms of memory and computation time.

[5]If a state must be controlled, it must be measurable and it enters the constraint $y \in \mathcal{Y}_k$.

Chapter 7

Markov Decision Processes

State-space representation of systems characterized by discrete states and by a non-linear update equation are better described by *Markov decision processes* (MDP). An MDP is a problem formulation for modeling sequential decisions guided by rewards under uncertainty in how the system transitions from state to state.

An MDP is characterized by

- a set $\mathcal{X}$ of N states;
- a set $\mathcal{U}$ of M control actions;

- the time-evolution described by a transition probability function[1]

  $$P : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \to [0, 1]$$

  where
  $$P^u_{x,x'} = \mathbb{P}[x'|x, u]$$

  is the *probability* to transition to state x' if the MDP is in state x and the input u is applied. The Markov property for probabilistic systems is the extension of the definition for deterministic systems: for every x', the probability $P^u_{x,x\prime}$ depends only on x and u and not on the past system history;

- an immediate cost
  $$r(x, u, x')$$

  for the transition to state x' given the MDP being in state x and when action u is taken.

MDP are typically represented by a Markov chain, with states depicted by circles connected by arrows with the corresponding transition probability.

[1] I increasingly like the notation from [3, Sect. 3.1] for the model dynamics

$$p : \mathcal{X} \times \mathcal{R} \times X \times \mathcal{U} \to [0, 1]$$

such that
$$p(x', r|x, u) \doteq Pr\{S_{t+1} = x', R_{t+1} = r|S_t = s, A_t = u\}.$$
The three argument function used in class is defined as

$$p(x'|x, u) = \sum_{r\in\mathcal{R}} p(x', r|x, u);$$

and the expected reward becomes

$$r(x', x, u) \doteq \mathbb{E}[R_t|S_t = x, A_t = u, S_{t+1} = x'] = \sum_{r\in\mathcal{R}} r\frac{p(x', r|x, u)}{p(x'|x, u)}.$$

The Bellman equation is

$$V^\pi_k(x) = \sum_u \pi_k(x, u) \sum_{x',r} p(x', r|x, u) \left[r + V^\pi_{k+1}(x')\right]$$

or, for the infinite horizon,

$$V^\pi(x) = \sum_u \pi_k(x, u) \sum_{x',r} p(x', r|x, u) \left[r + \gamma V^\pi(x')\right]$$

highlighting the quantity $r + \gamma V^\pi(x')$ similar to that in use in RL.

A control policy selects a suitable action $u \in \mathcal{U}$ based on the current state $x \in \mathcal{X}$ of the system. A policy can be deterministic

$$\mu : \mathcal{X} \to \mathcal{U}$$

or more generally, stochastic

$$\pi : \mathcal{X} \times \mathcal{U} \to [0,1].$$

where $\pi(x,u)$ is the conditional probability of selecting the input action u given that the MDP is in state x. A deterministic policy is a special case of a stochastic one, taking on the probability value 1 for the action u that the deterministic policy would select and 0 for all others. An optimal policy is deterministic[2], but for intermediate steps it is useful to work with stochastic policies.

The stage cost at time k for the states transition from x_k to x_{k+1} under the action u_k is

$$r_k = r(x_k, u_k, x_{k+1}).$$

The quantity we want to control is $\sum_{k=0}^{K} r_k$: in particular we want to find the policy $\boldsymbol{\pi} = \{\pi_0, \pi_1, \ldots \pi_K\}$ that minimizes

$$V^{\pi}(x) = \mathop{\mathbb{E}}_{x_0=x} \sum_{k=0}^{K} r_k$$

given the initial state $x_0 = x$.

7.1 Dynamic Programming for MDP

In order to solve this problem recursively, we define the accumulated cost

$$V_k^{\pi}(x) = \mathop{\mathbb{E}}_{x_k=x} \sum_{i=k}^{K} r_i,$$

[2]Explain why.

where the average is done on the policies $\{\pi_k, \pi_{k+1}, \dots \pi_K\}$. Since the system is Markovian, backward induction can be used to evaluate $V_k^\pi(x)$:

$$\begin{aligned}
V_k^\pi(x) &= \mathop{\mathbb{E}}_{x_k=x} \sum_{i=k}^{K} r_i \\
&= \mathop{\mathbb{E}}_{x_k=x} \left[r_k + \sum_{i=k+1}^{K} r_i \right] \\
&= \sum_u \pi_k(x,u) \sum_{x'} P^u_{x,x'} \left[r(x,u,x') + \mathop{\mathbb{E}}_{x_{k+1}=x'} \sum_{i=k+1}^{K} r_i \right] \qquad (*) \\
&= \sum_u \pi_k(x,u) \left[R^u_x + \sum_{x'} P^u_{x,x'} V^\pi_{k+1}(x') \right]
\end{aligned} \tag{7.1}$$

where

$$R^u_x = \sum_{x'} P^u_{x,x'} r(x,u,x')$$

is the averaged immediate cost.

To compute the average in $(*)$ we need to sum over the possible control actions u, each one of them having the probability $P^u_{x,x'}$ to transition the system from x to x'. The term in square brackets is the accumulated cost from time $k+1$ starting from the initial state $x_{k+1} = x'$ and the immediate cost $r(x,u,x')$ at k.

By Bellman's optimality principle, the optimal cost can be computed recursively using eq. (7.1) as a minimization on the action u

$$V_k^\star(x) = \min_\pi \sum_u \pi_k(x,u) \left[R^u_x + \sum_{x'} P^u_{x,x'} V^\star_{k+1}(x') \right].$$

For a finite horizon problem K and starting from the base case V_K, at each step k a policy π_k is found: under weak conditions[3], the policy is deterministic because the optimum is at one of the corners of the simplex defined by the constraints on the probabilities

$$\pi(x,u) \geq 0, \qquad \sum_u \pi(x,u) = 1.$$

Not sure about this one: the plot in class was given without labeling the axes.

[3]Which ones?

7.1.1 Complexity for Finite Horizon

We have reduced the original stochastic non-linear system to a linear one and solved by backward induction with linear programming. The price to pay is the increased size of the system of equations to solve: for each time step k, the minimization must be done M times, once for each control input u, for each of the N states; the policy π_k at step k reduces to a matrix of size $M \times N$ since the optimal policy is expected to be deterministic.

7.2 Infinite Horizon Problem

Most of the time when working with MDP we consider an infinite horizon, making the Markov process stationary with a time-invariant expected cost R_x^u and a time-invariant policy π.

The infinite-horizon cost is defined to be the quantity

$$\sum_{k=0}^{\infty} \gamma^k r_k \tag{7.2}$$

where $0 \leq \gamma < 1$ is the discount factor, which ensures that the cost is finite[4]. The value of γ achieves a balance between immediate and future costs and accounts for the lower impact of future costs: $\gamma \approx 0$ generates a myopic policy, $\gamma \approx 1$ a long-sighted one; for the extreme case $\gamma = 0$, only the immediate cost is considered.

The cost eq. (7.2) mirrors real-world problems: for instance it is economically advantageous to delay spending an amount of money that can instead be invested at a guaranteed base rate γ. It also maps to the Bernoulli termination problem (?), with $1 - \gamma$ being the termination probability.

7.2.1 Closed-Loop MDP

For a state-space system with linear dynamics $\dot{x} = Ax + Bu$, the feedback law $u = -Kx$ reduces the system dynamics to the closed loop response

$$\dot{x} = (A - BK)x.$$

[4]Without a discount factor the sum of the rewards can become infinite. Suppose that strategy π_1 results in a reward 1 for each time step and π_2 in a reward 100. A rational agent should prefer π_2 but both strategies have the same infinite cost [1].

Similarly, for an MDP, given the policy π, the dynamics reduces to a Markov chain where the closed-loop transition probabilities can be expressed by

$$P^{\pi}_{x,x'} = \sum_u \mathbb{P}[u|x]\mathbb{P}[x'|x,u] = \sum_u \pi(x,u)P^u_{x,x'}$$

Let $\mathtt{P}^{\pi} = \sum_u \pi(x,u)P^u_{x,x'} \in \mathbb{R}^{N\times N}$. Since $\mathtt{P}^{\pi}$ represents transition probabilities, its rows must sum to 1. Moreover, from the Perron-Frobenious, its eigenvalues have modulus 1.

At time k, the system will be in state x with probability distribution $d_k(x)$, which depends on π; the distribution evolves as the linear equation

$$d_{k+1}^{\top} = d_k^{\top}\mathtt{P}^{\pi}.$$

When the horizon is infinite, the stationary distribution $d(x)$ satisfies

$$d^{\top} = d^{\top}\mathtt{P}^{\pi}.$$

7.2.2 Example: The Stationary Distribution for the Traffic Light Problem

We consider the example problem of controlling a queue of vehicles at a traffic light. The states are represented by the number of cars waiting, the control actions are the states of the traffic light.

At each time step, there is a probability p that a new car arrives and $1-p$ that the queue length does not grow. We select the deterministic policy $\mu(x)$ that the traffic light turns green as soon as there are 3 cars waiting; we need not consider states with more than 3 cars because these states are never reached. Under this policy, the transition probabilities become

$$P^{\pi} = \begin{bmatrix} 1-p & p & 0 & 0 \\ 0 & 1-p & p & 0 \\ 0 & 0 & 1-p & p \\ 1-p & p & 0 & 0 \end{bmatrix}$$

The stationary distribution is found by solving the right eigenvector problem corresponding to the eigenvalue 1:

$$d = \frac{1}{3}\begin{bmatrix} 1-p \\ 1 \\ 1 \\ p \end{bmatrix}.$$

7.3 Computational Complexity of the Bellman Equation

The infinite horizon value of a policy can be expressed recursively similar to eq. (7.1) with no k dependence since policy and optimal cost are stationary

$$V^\pi(x) = \sum_u \pi(x,u) \left[R_x^u + \gamma \sum_{x'} P_{x,x'}^u V^\pi(x') \right] \tag{7.3}$$

and with the discount term γ. The proof is similar to that for eq. (7.1).

If the minimum is achieved at a deterministic policy, it becomes a system of N non-linear equations:

$$V^\star = \min_u \left[R_x^u + \gamma \sum_{x'} P_{x,x'}^u V^\star(x') \right].$$

We next consider two different approaches to solve this equation: policy iteration and value iteration.

7.3.1 Policy Iteration

Eq. (7.3) is a system of linear equations that can be written in the vector $\mathtt{V}^\pi \in \mathbb{R}^N$ as

$$\mathtt{V}^\pi = \mathtt{R}^\pi + \gamma \mathtt{P}^\pi \mathtt{V}^\pi \tag{7.4}$$

where $\mathtt{R}^\pi = \sum_u \pi(x,u) R_x^u$ is a N element vector.

Policy iteration starts by guessing a policy π_0 and then repeats the two steps until convergence of the policy:

- policy evaluation: compute the value function V^π associated to the policy π by solving eq. (7.4);
- policy update

$$\pi' = \arg\min_\nu \sum_u \nu(x,u) \left[R_x^u + \gamma \sum_{x'} P_{x,x'}^u V^\pi(x') \right] \tag{7.5}$$

An optimal policy can be indeed be found using the greedy approach of eq. (7.5), where the improved policy π' is only applied on one step only and the previous policy π is used for the computation of $V^\pi(x')$. This greedy improving finds the optimal policy because for any state x, it does not make the value function worse and it improves it for at least one state x'.

Theorem 4. *The policy update eq.* (7.5) *satisfies*

$$V^{\pi'}(x) \leq V^{\pi}(x)\ \forall x, \quad \text{and } \exists x' \text{ such that } V^{\pi'}(x) < V^{\pi}(x)$$

unless π is already the optimal policy.

Proof. If π' is the policy improving on π, one has

$$\begin{aligned}
V^{\pi}(x) &\geq \sum_u \pi'(x,u)\left[R_x^u + \gamma \sum_{x'} P_{x,x'}^u V^{\pi}(x')\right] \\
&= \mathbb{E}\left[r_0 + \gamma V^{\pi}(x_1)\right] \\
&\geq \mathbb{E}\left[r_0 + \gamma\left[r_1 + \gamma V^{\pi}(x_2)\right]\right] = \mathbb{E}\left[r_0 + \gamma r_1 + \gamma^2 V^{\pi}(x_2)\right] \\
&\geq \dots \\
&\geq \mathbb{E}\left[r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \dots\right] = V^{\pi'}(x)
\end{aligned}$$

Moreover the optimum is reached in a finite amount of improvements (at most $M \times N$), since there is a finite number of actions and states. □

7.3.2 Value Iteration

An alternative approach is to look for a fixed point of the Bellman optimality equation by iterating

$$V^{(t+1)}(x) = \min_{\pi} \sum_u \pi(x,u)\left[R_x^u + \gamma \sum_{x'} P_{x,x'}^u V^{(t)}(x')\right] \tag{7.6}$$

until convergence, *i.e.* when the *Bellman residual* falls below a certain threshold δ

$$\|V^{(t+1)} - V^{(t)}\|_{\infty} \leq \delta.$$

Convergence of the value iteration to $V^{\star}$ at rate γ is guaranteed by the observation that

$$||V^{(t+1)} - W^{(t+1)}||_{\infty} \leq \gamma ||V^{(t)} - W^{(t)}||_{\infty}$$

where $V^{(t)}$ and $W^{(t)}$ are two value functions. To see why the condition implies convergence, it is sufficient to take $W^{(t)}(x) = V^{\star}(x)$ identically, the fixed-point of eq. (7.6). Then $V^{(t)}(x)$ converges to $V^{\star}(x)$ at rate γ.

Value iteration starts by guessing a value function V_0 and then at each iteration updates the value function using eq. (7.6). At convergence, the optimal policy $\pi^{\star}$ is computed

$$\pi^{\star} = \arg\min_{\nu} \sum_u \nu(x,u)\left[R_x^u + \gamma \sum_{x'} P_{x,x'}^u V^{\star}(x')\right]$$

The computational complexity of value iteration is lower than policy iteration for iteration, because of the step required to solve the linear equation but the convergence is only asymptotic. For large discount factors γ with long horizon, the convergence is typically very slow.

Chapter 8

Monte-Carlo Learning

MDP require the knowledge of the system model. In the context of reinforcement learning, the value function V^π for the policy π however can be learned from *experience* by maintaining averages of the returns following each state: the average will converge to the state value $V^\pi(x)$ as the number of times that state is encountered approaches infinity [3, Sect. 3.5].

There are two settings for model-free methods:

- based on collected data, typically in the form of repeated episodes

  $$(x_0, u_0, r_0),\ (x_1, u_1, r_1), \ldots (x_T, u_T, r_T)$$

  for instance coming from repetition of the control task in a numerical simulator or controlled environment, *i.e.* experimentally. The episodes could be stochastic because the application of u_0 from state x_0 may transition the system to a different final state but the collected data will reveal only one realization;

- online during the control task with no prior training in a one-shot approach. The *adaptive control* approach is a challenging dual learn and control task and has been an open problem for decades.

The first approach is suitable for an incremental improvement of the policy; the second can deal with time-dependent models.

8.1 Q–Function

We have seen that $V^\pi(x)$ is a predictor of the quality of the candidate policy. For reinforcement learning where the learning comes from experimental data, it is more convenient to reformulate the problem in terms of the quality Q function

$$Q^\pi(x,u) = R_x^u + \gamma \sum_{x'} P_{x,x'}^u V^\pi(x'). \tag{8.1}$$

$Q^\pi(x,u)$ is the expected future cost when taking action u in state x, as dictated by the policy π. In contrast to the value function $V(x) \in \mathbb{R}^N$, the size of Q is $N \times M$: this increased problem size has the advantage, amongst others, to make the policy update step of the policy iteration trivial.

The value function is the expected Q function averaged over the policy π

$$V^\pi(x) = \sum_u \pi(x,u) Q^\pi(x,u). \tag{8.2}$$

The Bellman equation can be written in terms of the Q function in closed form using eq. (8.2) in eq. (8.1)

$$Q^\pi(x,u) = R_x^u + \gamma \sum_{x'} P_{x,x'}^u \sum_{u'} \pi(x',u') Q^\pi(x',u') \tag{8.3}$$

and the Bellman optimality principle as

$$Q^\star(x,u) = R_x^u + \gamma \sum_{x'} P_{x,x'}^u \left(\min_{u'} Q^\star(x',u') \right) \tag{8.4}$$

where here we used the previous observation that the optimal policy is deterministic.

8.1.1 Policy Iteration for the Q function

The policy iteration with the Q function is rewritten as

- the *policy evaluation* requires solving the higher-dimensional linear system eq. (8.3);
- the *policy update* becomes even simpler than for the value function

$$\pi(x,u) = \arg\min_\nu \sum_u \nu(x,u) Q(x,u).$$

Note that now only the policy evaluation requires the model information which is contained in the large matrix $Q(x,u)$.

We are now left with the task of computing the policy assessment step: the advantage of this approach is that this can be obtained from the simulated data:

- let the system be controlled by a policy π and collect a sufficiently long episode
$$(x_0, u_0, r_0),\ (x_1, u_1, r_1), \dots (x_T, u_T, r_T);$$

- compute the *empirical cost* g_0, g_1,… g_T from the immediate costs r_i and the discount cost factor γ
$$g_k = \sum_{i=k}^{T} \gamma^{i-k} r_i; \tag{8.5}$$

- interpret g_k as a realization of the return[1] of the state-action pair (x_k, u_k)
$$Q^\pi(x_k, u_k) \approx g_k.$$
If the system is stochastic and therefore governed by transition probabilities, $Q^\pi(x, u)$ must be computed as the average over multiple visits
$$Q^\pi(x, u) = \frac{\sum_{t=1}^{T} 1[x_t = x, u_t = u] g_t}{\sum_{t=1}^{T} 1[x_t = x, u_t = u]} \tag{8.6}$$
where the accumulation must be sufficiently long to see all possibles combinations of states and inputs.

This method assumes no explicit knowledge of transition probabilities $P^u_{x,x'}$.

A few remarks are required:

- the policy cannot be evaluated, and therefore Q cannot be computed, until the episode is terminated. This is because the empirical cost eq. (8.5) requires one to use the same policy until the end of the episode. MC learning method is therefore not adequate to adaptive learning;

- there is no guarantee that the *estimate* of Q, *i.e.* for a single realization, eq. (8.6), satisfies the self-consistent Bellman equation eq. (8.3). The identity is however valid in the limit of infinite realizations;

- Markovianity? I did not understand the remark in the lecture.

[1]By definition, $Q^\pi(x, u)$ is the cost of applying input u when in state x and then applying the policy π.

8.1.2 Exploration vs Exploitation

The greedy policy improvement step can be stuck in a suboptimal solution when the triple (x, u, r) is never explored by the policy and the associated Q value is not available. This is usually fixed by *e.g.*

- the ϵ-greedy policy improvement, where a random action from all possible actions $\mathcal{U}$ is tried with probability ϵ

  $$\pi(x,u) = \begin{cases} \arg\min_u Q(x,u) & \text{with probability } 1-\epsilon \\ \text{Uniform}(\mathcal{U}) & \text{with probability } \epsilon \end{cases}. \tag{8.7}$$

  A large ϵ allows for more exploration; or

- the Bolzmann policy improvement[2]

  $$\pi(x,u) = \frac{e^{-\beta Q(x,u)}}{\sum_u e^{-\beta Q(x,u)}}.$$

  Here a small β implies more exploration.

In both cases, if done "properly", each state-action pair is visited infinitely many times and the policy converges to a greedy policy with probability 1.

8.2 Scalability and Parametrization of Q

The computational complexity of the Q function becomes intractable when the state-action space is large, either because the system has intrinsically many states and inputs or as a result of the discretization of a continuous system. The trick is to parametrize the Q function by a small set of basis functions

$$Q_\theta(x,u), \quad \theta \in \mathbb{R}^d \quad \text{where } d \ll MN$$

thereby restricting unnecessary degrees of freedom. A smart parametrization may allow one to guess the Q function also for state-action pairs that have not been observed. For continuous systems the basis functions are continuous functions and the summation in the Bellman equation must be replaced by integrals; nevertheless the parametrization transforms the initial system into a discrete one.

[2]A note on the initialization of the Q function: the Bolzmann policy requires the knowledge of its values also for state-action pairs that have not been explored. If the optimization is set up to minimize positive costs, Q is initialized to a large number; in the case of maximization, as it is more common in RL settings, then 0 is commonly used.

Prior information on the problem may suggest a smart parametrization. For instance in the LQR case, we assumed that the value function is quadratic in x and u (?). Did not understand the remark from the lecture.

The downside of the parametrization is that the optimal $Q^\star$ may not belong to the lower dimensional space. Therefore the solution to the Bellman equation for a given policy π may not belong to the space

$$\mathcal{Q} = \{Q_\theta,\ \theta \in \mathbb{R}^d\}$$

and policy iteration may not converge or converge to the suboptimal solution. Moreover the policy update step may require additional computation that in the discrete case was limited to selecting one entry in the Q function.

8.2.1 Linear Parametrization

We consider the linear parametrization

$$Q_\theta(x,u) = \sum_{\ell=1}^{d} \phi_\ell(x,u)\theta_\ell = \Phi(x,u)\theta \tag{8.8}$$

where $\phi_\ell(x,u)$ are basis functions. These can be

- polynomials $\{1, x, u, x^2, u^2, xu, \ldots\}$;
- quadratic forms $x^\top A_\ell x$ where A_ℓ can be a matrix with only one element different from zero or a set of positive definite matrices;
- a set of discrete basis functions[3].

In general the solution to the Bellman equation eq. (8.3) is not in the form $\Phi\theta$ and the equation can only be solved approximately

$$Q_\theta(x,u) \approx \underbrace{R_x^u + \gamma \sum_{x'} P_{x,x'}^u \sum_{u'} \pi(x',u') Q_\theta(x',u')}_{\doteq \mathcal{B}_\pi(Q_\theta)}$$

where, in matrix form,

$$\mathcal{B}_\pi(Q) = R + \gamma T^\pi Q. \tag{8.9}$$

[3]In the game of Tetris on a board size of 20×10, there are at least 2^{200} states, each board position being either empty or occupied. A solution mimicking the human approach has been realized with $d = 21$ only.

The *projected Bellman equation* seeks a solution Q_θ by first projecting $\mathcal{B}(Q_\theta)$ into the subspace $\mathcal{Q}$ using the projection operator

$$\Pi(\mathcal{B}(Q_\theta)) = \arg\min_{Q_{\hat{\theta}}} \left\| Q_{\hat{\theta}} - \mathcal{B}(Q_\theta) \right\|_\rho^2$$

in the ρ-norm[4] and then looks for the fixed point

$$Q_\theta = \Pi(\mathcal{B}(Q_\theta)). \tag{8.10}$$

The matrix ρ is a tuning parameter (as long as it is positive definite otherwise it does not define a norm) and may be taken to weight the states differently according to their frequency or because they are of particular interest [3, Sect. 11.4].

Theorem 5. *The fixed point of the projected Bellman equation is given by*

$$\Phi^\top \rho \left(I - \gamma T^\pi\right) \Phi\theta = \Phi^\top \rho R. \tag{8.11}$$

Proof. Let $Q_\theta = \Phi\theta$; by eq. (8.9) we have $\mathcal{B}(Q_\theta) = R + \gamma T^\pi \Phi\theta$. To find $\Pi(B(Q_\theta))$, we compute

$$\arg\min_{\hat{\theta}} \frac{1}{2} \left\| \Phi\hat{\theta} - R - \gamma T^\pi \Phi\theta \right\|_\rho^2$$

by setting the gradient with respect to $\hat{\theta}$ to zero, to give

$$\Phi^\top \rho \Phi\hat{\theta} = \Phi^\top \rho \left(R + \gamma T^\pi \Phi\theta\right).$$

and $\Pi(B(Q_\theta)) = \Phi\hat{\theta}$. Imposing the fixed point condition eq. (8.10) gives

$$\Phi\hat{\theta} = \Phi\theta$$

which we can substitute into the equation above. After rearranging the terms, we obtain eq. (8.11). □

Eq. (8.11) requires the model information contained in T^π. One can however contruct the matrices $\Phi^\top \rho\Phi$, $\Phi^\top \rho R$ and $\Phi^\top \rho T^\pi \Phi$ from the episodes/trajectories, and so have a model free approach, by making use of the freedom to choose ρ. If we choose ρ to represent the relative frequency that the state-action

[4]The ρ-norm is defined as

$$\|v\|_\rho^2 = v^\top \rho v.$$

pair (x, u) appears in the episode, from the data and the basis function, one can construct the required terms as

$$\begin{aligned}
\Phi^\top \rho \Phi &\approx \frac{1}{K} \sum_{k=1}^{K} \phi(x_k, u_k)^\top \phi(x_k, u_k) \\
\Phi^\top \rho R &\approx \frac{1}{K} \sum_{k=1}^{K} \phi(x_k, u_k)^\top r_k \\
\Phi^\top \rho T^\pi \Phi &\approx \frac{1}{K} \sum_{k=1}^{K} \phi(x_k, u_k)^\top \phi(x_{k+1}, u_{k+1}).
\end{aligned}$$

Chapter 9

Reinforcement Learning

We have seen how the policy and value iteration methods require either a model or a full and sufficiently rich trajectory. MC learning forces us to wait until the end of the episode to improve the policy. In this Section, we want to learn the system as the data is collected along a single trajectory while also controlling the system.

In MC learning, to construct the Q function from specific realizations, we could have not made use of Bellman equation, eq. (8.3) because this is only satisfied in the limit of infinitely long episode. For a finite episode, the *temporal difference error* is the amount by which Bellman equation is not satisfied,

$$e = r + \gamma Q(x', u') - Q(x, u) \tag{9.1}$$

for two subsequent data points

$$(x, u, r) \quad (x', u').$$

In general the error e is not zero but its average must be, $\mathbb{E}[e] = 0$, according to the Bellman equation.

An iterative way to find the Q function that solves $\mathbb{E}[e] = 0$ is provided by the stochastic approximation. Let q a decision variable (?) and $e(q)$ a random variable for which we want to find $q^\star$ that solves $\mathbb{E}[e(q^\star)] = 0$. Then the iteration[1]

$$q_{k+1} \leftarrow q_k - \alpha_k e(q_k) \tag{9.2}$$

converges to the solution $q^\star$ of $E[e(q)] = 0$ provided

- $e(q)$ is bounded;

[1]Why the $\leftarrow$ and not simply $=$? This is how it was written in the lecture, asked on Moodle, no answer yet.

- $\mathbb{E}[e(q)]$ is non-decreasing in q;
- $\mathbb{E}(e'(q^\star))$ exists and is positive;
- the sequence α_k satisfies

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty$$

The iteration is a gradient-like update with a time-varying time gain α_k that can be taken to be $\alpha_k = \frac{1}{k}$. At each iteration, q is updated using a statistically decreasing (in absolute value) contribution $\alpha_k e(q_k)$: is α_k decreasing too slowly, the convergence to $q^\star$ is slow (in the case $\alpha_k = \alpha$ is a constant, no convergence occurs); is α_k decreasing too fast (*e.g.* $\alpha_k = \frac{1}{k^2}$), the sequence $\{q_k\}$ converges to a wrong value that depends on the initial value of the sequence.

A better intuition of how this algorithm works would be a good idea.

9.0.1 TD Learning: SARSA

The SARSA algorithm[2] uses the empirical evaluation of the TD error eq. (9.1) to update the Q function via the stochastic approximation iteration eq. (9.2)[3]

$$Q(x,u) \leftarrow Q(x,u) + \alpha_k \left[r + \gamma Q(x',u') - Q(x,u) \right] \tag{9.3}$$

The SARSA algorithm interleaves policy evaluation with policy improvement: at every time k, it

- selects u via an ϵ-greedy policy eq. (8.7) based on the latest estimate of $Q(x,u)$; and
- performs the iterative update eq. (9.3) using a constant $\alpha_k = \alpha$ (which would lead to a steady-state non-zero variance but it is suitable for non-stationary problems) while taking $\epsilon \to 0$ as $k \to \infty$ to control the convergence of the learning.

Contrast this with MC learning, where the algorithm needs to wait until the end of the episode to estimate Q.

[2] SARSA derives its name from the use in the reinforcement learning community of the tuple

$$(s, a, r, s', a')$$

for the update iteration where s and a are the state and action and s' and a' the next state and action.

[3] Why is the sign $+$? In the notes the index k refers to both the state and the iteration number but α is not state-dependent. I remove k for the states.

9.0.2 TD Learning: Q learning

Value iteration solves the Bellman equation without a policy improvement step. We can apply TD learning directly to Bellman optimality principle because in eq. (8.4), the optimal action u is the one that minimizes $Q^\star(x', u)$.

In our case, we hope we can select such u by maintaining an *estimate* of the optimal Q function that is continuously updated from the individual realizations

$$Q^\star(x,u) \approx r + \gamma \min_{u'} Q^\star(x', u')$$

because the RHS is the same as the RHS of eq. (8.4) in the limit of an infinitely long episode:

$$Q^\star(x,u) = \mathbb{E}_{r,x'}\left[r + \gamma \min_{u'} Q^\star(x', u')\right].$$

The Q learning algorithm updates the estimate of the Q function using the stochastic approximation[4]

$$Q(x,u) \leftarrow Q(x,u) + \alpha_k \left(r + \gamma \min_{u'} Q(x', u') - Q(x,u)\right). \tag{9.4}$$

There are a few things to notice:

- the next input u' does not appear in eq. (9.4) and the Q update is independent on the action used. Q learning is an *off-policy algorithm* because it learns the best policy even from a random input;
- convergence to the optimal Q is guaranteed under the same assumptions as for the stochastic approximation;
- the policy is typically updated as ϵ-greedy;
- for large state and input spaces, one usually approximate the Q function with linear approximants or neural networks;
- the stochastic approximation step eq. (9.4) is *forward* dynamic programming, going forward in time.

[4] Is a_k also here taken as a constant?

9.0.3 Q Learning with Linear Approximant

The lecture is not clear: I believe the loss function is a projection operation but the question was not answered in the forum.

The approach of Sec. 8.2.1 can be applied here too:

- approximate Q by the low-dimensional linear combination eq. (8.8);
- minimize the loss function

  $$\min_\theta \underbrace{\frac{1}{2}\left\|\Phi(x,u)\theta - Q^+\right\|^2}_{L(\theta)}$$

  where

  $$Q^+ = R_x^u + \gamma \sum_{x'} P_{x,x'}^u \left(\min_{u'} \Phi(x',u')\theta^\star \right)$$

  since the Bellman optimality principle will not in general have a solution in the space spanned by Φ. (Where is the equation coming from? From the projection into the $\mathcal{Q}$ space?)

A solution to the minimization is found iteratively[5], by using a gradient descent approach

$$\theta \leftarrow \theta - \alpha \nabla L(\theta) = \theta - \alpha \left(\Phi^\top \theta - Q^+ \right) \Phi$$

Similarly to what done in Q learning, we use the individual realization

$$Q^+ \approx r + \gamma \min_{u'} \Phi(x',u')\theta$$

as an estimate for Q^+.

[5]Why not solving the linear problem directly? It does not work when the space is too large?

List of References and Additional Readings

Here is a list of references that I (Michele Zaffalon) found useful to understand the class.

- *Model Predictive Control: Theory, Computation, and Design* by Rawlings, Mayne and Diehl [2]. This is an introductory book.
- *Algorithms for Decision Making* by Kochenderfer, Wheeler and Wray [1]. A Julia-based introduction to algorithms for decision making under uncertainty with a practical approach. Monte-Carlo and RL (SARSA and Q-learning) are described as well as more advanced techniques.
- *Reinforcement Learning, An Introduction* by Sutton and Barto [3]. A classic referenced book that expands on the MDP, MC and RL lectures giving an intuition without delving too much into the mathematics in a verbose way.

Bibliography

[1] *Algorithms for Decision Making*. 1st edition. Available at `https://algorithmsbook.com/`. MIT Press, 2022.

[2] *Model Predictive Control: Theory, Computation, and Design*. 2nd edition. Available at `https://sites.engineering.ucsb.edu/~jbraw/mpc/`. 2022.

[3] *Reinforcement Learning, An Introduction*. 2nd edition. Available at `http://incompleteideas.net/book/RLbook2020.pdf`. MIT Press, 2020.