CHEM-E7225 Advanced Process Control D — Winter 2026

Lecture 05 – Model Predictive Control (A)

Otacilio "Minho" Neto (otacilio.bneto@pm.me)

The previous lectures introduced important concepts from *dynamic systems theory* and *(convex) optimization theory*. Finally, we are ready to combine these two theories into our goal application: the optimal control of dynamic systems. In this lecture, we start with the key component of optimal controllers, namely, the optimal control problem.

1 Optimal control problems (OCP) – General formulations

A generic *discrete-time finite-horizon optimal control problem* (OCP) is the constrained nonlinear program

$$\underset{x[\cdot],\ u[\cdot]}{\text{minimize}} \quad \sum_{k=0}^{N-1} L(x[k],u[k]) + V_f(x[N]) \tag{1a}$$

$$\text{subject to} \quad x[k{+}1] = f_{\Delta t}(x[k],u[k]), \qquad k=0,\dots,N{-}1, \tag{1b}$$

$$h(x[k],u[k]) \preceq 0, \qquad k=0,\dots,N{-}1, \tag{1c}$$

$$b(x[0],x[N]) = 0. \tag{1d}$$

The *decision vectors* are both the finite sequence $\mathbf{x} = (x[0],\dots,x[N])$ and $\mathbf{u} = (u[0],\dots,u[N{-}1])$, for a total of $(N{+}1)N_x + NN_u$ decision variables. The *objective function* (Eq. 1a) is defined by the *stagewise loss* $L(\cdot)$ and the *terminal value* $V_f(\cdot)$ functions. The *dynamic constraints* (Eq. 1b) are N vector-valued equality constraints that enforce the system dynamics, according to the transition function $f_{\Delta t}(\cdot)$ (here, assumed to be a time-discretization of an ODE model $f(\cdot)$ using a period $\Delta t > 0$). The *path constraints* (Eq. 1c) are N vector-valued inequality constraints that enforce desired and/or mandatory requirements (actuator limits, safety constraints, physical restrictions, etc.). The *boundary conditions* (Eq. 1d) are vector-valued equality constraints that used to enforce initial and terminal conditions. Finally, $N > 0$ is the *control horizon* (corresponding to the total number of control actions).

The OCP (1) is not the most general, but already comprises a wide range of relevant problems. Importantly, it is a *finite-horizon* program, thus well-suited for batch or short-term operations (its extension to continuous operation will be discussed later). Our task is to define the control horizon N and functions $\{L, V_f, h, b\}$, such that a solution

$$(\mathbf{x}^\star, \mathbf{u}^\star) \in \arg\min \left\{ V_N(\mathbf{x},\mathbf{u}) \mid (\mathbf{x},\mathbf{u}) \in \mathcal{Z}_N \right\}, \tag{2}$$

consists of the *optimal control actions* $\mathbf{u}^\star = (u^\star[0],\dots,u^\star[N{-}1])$ and *optimal state responses* $\mathbf{x}^\star = (x^\star[0],\dots,x^\star[N])$, with respect to some desired higher-level goal (maximize production, minimize energy costs, etc.). We remark that $\mathbf{x}^\star$ is an optimal *prediction*, and that the actual response of the physical system might differ.

Remark. *Problem* (1) *arises from employing* direct multiple-shooting *to solve an underlying continuous-time control problem (as the original model is continuous). There are other* direct methods *(single-shooting, collocation, etc.), each resulting in slightly different OCPs. Unfortunately, we cannot cover all of them in this course (see [1, §8.5] for more).*

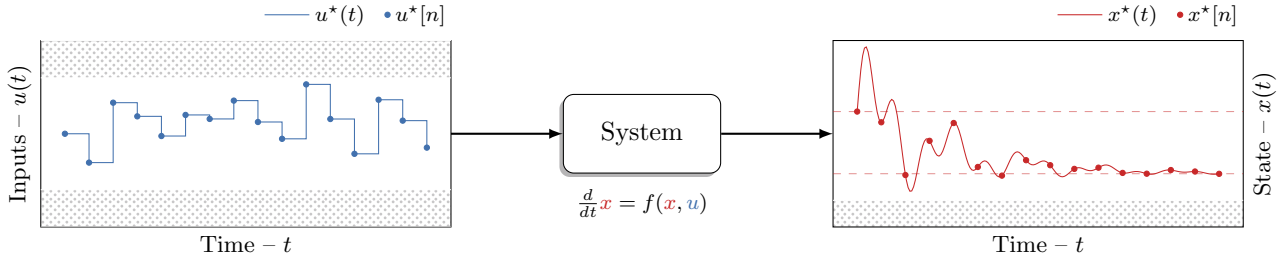

Figure 1. Illustration of a solution of an optimal control problem. The optimal controls $\mathbf{u}^\star = (u^\star[0],\dots,u^\star[N{-}1])$ and state responses $\mathbf{x}^\star = (x^\star[0],\dots,x^\star[N])$ minimize some relevant metric (e.g., distance to a setpoint). The state evolution is related to the controls according to the dynamical model $f_{\Delta t}(\cdot)$, as enforced by the dynamic constraints. The shaded regions indicate unfeasible controls/states according to the path constraints $h(x[k],u[k]) \preceq 0$ $(\forall k)$. The dashed lines in the right plot illustrate the boundary conditions $b(x[0],x[N]) = 0$.

Example 1.1 ***Setpoint tracking with input saturation limits***

Problem statement:
The state $x(t)$ of a dynamical system must be driven to a constant setpoint $r(t) = x_{\text{ref}}$, while applying the least amount of control effort (with $u(t) = 0$ being the zero-effort control). The setpoint tracking must be achieved within a $T = 1$ [units-of-time] interval, assuming that the system is initially at position $x(0) = x_0$. Finally, this must be achieved while satisfying the input saturation limits $-10 \leq u(t) \leq 10$ $(\forall t)$.

Optimal control design:
Firstly, we fix a *control frequency* $1/\Delta t$, leading to the problem having a *control horizon* of $N = T/\Delta t$ actions. We should now design a quantitative metric that, if minimized, makes the system closer to the setpoint without too much control effort. An option is to quantify *"distance to the desired point at each kth timestep"*, which can be expressed for each $k = 1, \ldots, N-1$ as the (often conflicting) quadratic objectives

$$\underset{(x[k] - x_{\text{ref}})^2}{\text{[Setpoint distance]}} \quad \underset{(u[k] - 0)^2}{\text{[Control energy]}} \tag{3}$$

Simultaneously accounting for all timesteps, we obtain the control objective

$$V_N(\mathbf{x}, \mathbf{u}) = \sum_{k=0}^{N-1} \underbrace{\left(\alpha(x[k] - x_{\text{ref}})^2 + \beta u[k]^2\right)}_{L(x[k],u[k])} + \underbrace{\alpha_F(x[N] - x_{\text{ref}})^2}_{V_f(x[N])}, \tag{4}$$

in which $\{\alpha, \alpha_F, \beta\}$ are positive tuning parameters that control the trade-off between matching the setpoints (α, α_F) and saving control effort (β). Geometrically, minimizing this objective means that we want to minimize the area below the curves $\alpha(x(t) - x_{\text{ref}})$ and $\beta u(t)$, as illustrated in Figure 2.

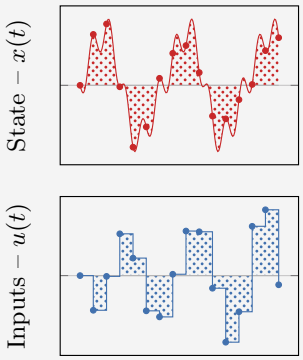

Figure 2

After formalizing the control objective, we must now write all relevant constraints. Let's assume that the transition function is an Euler integrator of the continuous-time dynamics, that is, $f_{\Delta t}(x[k], u[k]) = x[k] + f(x[k], u[k])\Delta t$ for all $k = 0, \ldots, N-1$. The path constraints are simply the input saturation limits, that is, $-10 \leq u[k] \leq 10$ for all $k = 0, \ldots, N-1$. Finally, the boundary conditions are only the initial state constraint, $x[0] = x_0$.

Putting all together, an optimal control problem that formalizes the original task is

$$\underset{x[\cdot],\ u[\cdot]}{\text{minimize}} \quad \sum_{k=0}^{N-1} \left(\alpha(x[k] - x_{\text{ref}})^2 + \beta u[k]^2\right) + \alpha_F(x[N] - x_{\text{ref}})^2 \tag{5a}$$

$$\text{subject to} \quad x[k{+}1] - x[k] = f(x[k], u[k])\Delta t, \quad k = 0, \ldots, N-1, \tag{5b}$$

$$-10 \leq u[k] \leq 10, \quad k = 0, \ldots, N-1, \tag{5c}$$

$$x[0] = x_0 \tag{5d}$$

which is a convex QP if $f(\cdot)$ is linear (and a nonconvex QP otherwise).

The example illustrates two important lessons: (i) Different higher-level goals (i.e., setpoint tracking and control effort minimization) admit similar formulations (i.e., quadratic functions); and (ii) Multiobjective problems require tuning parameters to quantify the relative importance of each goal. However, the most important fact illustrated by the above example is that *general problems can be cast into specific class of optimization programs*. In the following, we discuss a fundamental class of OCPs which formalizes (or approximates) most of the problems we care about.

2 Linear quadratic regulator (LQR)

The *linear quadratic regulator* (LQR) problem refers to the convex QP

$$\underset{x[\cdot],\ u[\cdot]}{\text{minimize}} \quad \frac{1}{2}\sum_{k=0}^{N-1}\begin{bmatrix}x[k]\\u[k]\end{bmatrix}^\mathsf{T}\begin{bmatrix}Q & S\\ S^\mathsf{T} & R\end{bmatrix}\begin{bmatrix}x[k]\\u[k]\end{bmatrix} + \frac{1}{2}x[N]^\mathsf{T}Q_f x[N] \tag{6a}$$

$$\text{subject to} \quad x[k{+}1] = Ax[k] + Bu[k], \quad k = 0,\dots,N{-}1, \tag{6b}$$

$$x[0] = x_0, \tag{6c}$$

given *tuning matrices* (Q, R, S, Q_f) and *system matrices* (A, B), with nonnegative $Q, Q_f \succeq 0$ and positive $R \succ 0$. The underlying control objective is to regulate a linear system around the origin $(x_{\text{ref}}, u_{\text{ref}}) = (0, 0)$. Despite innocent-looking, the LQR problem is a fundamental OCP for many reasons. Firstly, we already know that nonlinear systems can be approximated by linear models (via linearization), and that we can always shift a state-space such that the origin corresponds to some desired steady-state. Therefore, it covers any control problem of driving a system (linear or not) to a setpoint, assuming there are no path constraints. It also arises as a subroutine of more advanced control methods. Finally, not least importantly, it has a widely-known and well-structured closed-form solution–allowing for embedded implementations with very-high control frequencies.

Problem data / Controller tuning

The LQR Problem (6) is a QP, and, as such, fully defined by the tuning (Q, R, S, Q_f) and system (A, B) matrices. The diagonal entries of the tuning matrices quantify the importance of the corresponding state/control variables, since

$$\begin{bmatrix}x_1\\x_2\\\vdots\\x_{N_x}\end{bmatrix}^\mathsf{T}\begin{bmatrix}q_{11} & q_{12} & \cdots & q_{1N_x}\\ q_{21} & q_{22} & \cdots & q_{2N_x}\\ \vdots & \vdots & \ddots & \vdots\\ q_{N_x1} & q_{N_x2} & \cdots & q_{N_xN_x}\end{bmatrix}\begin{bmatrix}x_1\\x_2\\\vdots\\x_{N_x}\end{bmatrix} = \sum_{i=1}^{N_x} q_{ii}x_i^2 + 2\sum_{i=1}^{N_x}\sum_{j=i+1}^{N_x} q_{ij}x_ix_j,$$

and similarly for $u^\mathsf{T}Ru$ and $x^\mathsf{T}Su$. Thus, it is very common that the diagonal entries of (Q, R) are the main tuning parameters: selecting $Q \succ R$ (i.e., the diagonal entries of Q are larger than those of R) leads to fast stabilization with large control efforts, while tuning $Q \prec R$ leads to control actions of smaller effort at the expense of slower stabilization. The terminal tuning matrix Q_f is particularly important. While $Q_f = Q$ is a common assumption, it can be shown that the LQR Problem exhibit some desirable stability/robustness properties if Q_f is chosen to be the solution to

$$Q_f = Q + A^\mathsf{T}Q_fA - (S + A^\mathsf{T}Q_fB)(R + B^\mathsf{T}Q_fB)^{-1}(S^\mathsf{T} + B^\mathsf{T}Q_fA), \tag{7}$$

for reasons that will become clear later. The above is known as the *discrete-time algebraic Riccati equation* (DARE). The matrices (A, B) are, of course, determined by the dynamics of the system to be controlled.

There are no universal tuning rules for (Q, R, S)–they are problem dependent. The *cross-term tuning matrix* S is commonly chosen to be zero, $S = 0$, as there are only a few control problems in which $x^\mathsf{T}[k]Su[k]$ has any operational meaning. The *control tuning matrix* R is commonly chosen to be diagonal, $R = \text{diag}(r_1, \dots, r_{N_u})$, given N_u individual positive weighting factors. The *state tuning matrix* Q is commonly tuned as follows: Assuming we actually want to regulate a *performance signal* modelled as $z = Cx$, we can choose

$$Q = C^\mathsf{T}Q_zC + \rho I, \qquad \text{given } Q_z \succ 0, \quad \rho > 0.$$

Note that, due to this choice, the term $x^\mathsf{T}Qx$ becomes $z^\mathsf{T}Q_zz + \rho x^\mathsf{T}x$. This tuning rule has two advantages. First, it translates the choice of N_x^2 parameters to N_z^2 parameters (thus simplifying the tuning, as $N_z \ll N_x$ in most problems). Secondly, the regularization term $\rho x^\mathsf{T}x$ ensures that Q is positive; leading to the LQR problem being *strictly convex*.

2.1 Solving the LQR Problem

Batch approach

A solution to the LQR Problem (6) can be obtained by rewriting the program into standard QP form, then using the KKT conditions (that is, the gradients of its Lagrangian) to obtain a solution via root-finding methods. Since the problem is an equality-constrained QP, the root-finding problem is characterized by a linear *KKT System* (see Lecture Notes L04). In the LQR Problem, with a slight reordering of the *KKT Matrix*, this system takes the form

$$\begin{bmatrix} \partial L(\cdot)/\partial \lambda[0] \\ \partial L(\cdot)/\partial x[0] \\ \partial L(\cdot)/\partial u[0] \\ \partial L(\cdot)/\partial \lambda[1] \\ \partial L(\cdot)/\partial x[1] \\ \partial L(\cdot)/\partial u[1] \\ \partial L(\cdot)/\partial \lambda[2] \\ \vdots \\ \partial L(\cdot)/\partial \lambda[N] \\ \partial L(\cdot)/\partial x[N] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \iff \begin{bmatrix} I & & & & & & & & & \\ & Q & S & A^\mathsf{T} & & & & & & \\ & S^\mathsf{T} & R & B^\mathsf{T} & & & & & & \\ & A & B & & -I & & & & & \\ & & & -I & Q & S & A^\mathsf{T} & & & \\ & & & & S^\mathsf{T} & R & B^\mathsf{T} & & & \\ & & & & A & B & & & & \\ & & & & & & \ddots & \ddots & B^\mathsf{T} & \\ & & & & & & & B & & -I \\ & & & & & & & & -I & Q_f \end{bmatrix} \begin{bmatrix} \lambda^\star[0] \\ x^\star[0] \\ u^\star[0] \\ \lambda^\star[1] \\ x^\star[1] \\ u^\star[1] \\ \lambda^\star[2] \\ \vdots \\ \lambda^\star[N] \\ x^\star[N] \end{bmatrix} = \begin{bmatrix} x_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}. \tag{8}$$

Despite large, note that the above system has a well-defined *structure*. Specifically, it is defined by a *block tridiagonal matrix*, allowing us to solve it efficiently by banded factorization methods. Additionally, the structure of this KKT System is such that there are methods tailored *specifically* for it, as discussed in the next subsection.

Of course, very rarely we need to formulate and solve the KKT System shown above–instead, modern solvers allow us to write the OCP directly as formulated in Eq. (6), and then they solve the system for us. However, it is still important to recognize the above structure and the fact that it brings computational benefits.

Dynamic programming approach

An alternative method, which is both elegant and efficient, to solve the LQR Problem (6) is the *dynamic programming* approach. It is based on the observation that the optimal value of a stagewise OCP can be written as the recursion

$$V_n(x[n]) = \min_{\mathbf{u}} \left(\sum_{k=n}^{N-1} L(x[k], u[k]) + V_f(x[N]) \right) \tag{9}$$

$$= \min_{u[n]} \left\{ L(x[n], u[n]) + \underbrace{\min_{\mathbf{u}} \left(\sum_{k=n+1}^{N-1} L(x[k], u[k]) + V_f(x[N]) \right)}_{V_{n+1}(x[n+1])} \right\} \tag{10}$$

for each $n = 0, \dots, N-1$, with terminal $V_f(x[N])$. The function $V_n(x[n])$ is the nth step *value function* or *cost-to-go*, as it represents the total (optimal) cost of "going" to the state $x[n]$. The outer minimization problem above is an unconstrained nonlinear program, which we can solve directly using the *stationarity condition*. Therefore, solving an optimal control problem (w/ no path constraints) is equivalent to solving the root-finding problem

$$\text{Find } u^\star[n] \in \mathbb{R}^{N_u} \text{ such that } \frac{\partial}{\partial u^\star[n]} \Big(L(x[n], u^\star[n]) + V_{n+1}\big(x[n+1]\big) \Big) = 0, \tag{11}$$

for each $n = 0, \dots, N-1$, in which the state $x[n]$ is given and $x[n+1] = f_{\Delta t}(x[n], u^\star[n])$. We remark that the above observation is very powerful, since it gives us a principled approach to solve **any** OCP. The only (and big!) challenge is that we have to know the value functions $(V_0, \dots, V_N)$ in closed-form, which is not possible in most cases.

Remark. *Many algorithms in the popular field of* reinforcement learning *(RL) are based on approximating $V_N(\cdot)$ using real-time data (the action-reward principle). That's why they are also called* approximate dynamic programming.

In linear-quadratic (LQ) problems, the dynamic programming approach is readily applicable. The reason is that LQ problems are known to have quadratic value functions, that is, $V_n(x[n]) = x[n]^\mathsf{T} P_n x[n]$ for some matrix $P_n \succeq 0$.

Combining this with $L(\cdot)$ being quadratic and $f_{\Delta t}(\cdot)$ being linear, the stationarity condition in Eq. (11) becomes

$$\frac{\partial}{\partial u^\star[n]} \left(\begin{bmatrix} x[n] \\ u^\star[n] \end{bmatrix}^\mathsf{T} \begin{bmatrix} Q & S \\ S^\mathsf{T} & R \end{bmatrix} \begin{bmatrix} x[n] \\ u^\star[n] \end{bmatrix} + (Ax[n] + Bu^\star[n])^\mathsf{T} P_{n+1} (Ax[n] + Bu^\star[n]) \right) = 0,$$

$$\Rightarrow \frac{\partial}{\partial u^\star[n]} \left(\begin{bmatrix} x[n] \\ u^\star[n] \end{bmatrix}^\mathsf{T} \begin{bmatrix} Q + A^\mathsf{T} P_{n+1} A & S + A^\mathsf{T} P_{n+1} B \\ S^\mathsf{T} + B^\mathsf{T} P_{n+1} A & R + B^\mathsf{T} P_{n+1} B \end{bmatrix} \begin{bmatrix} x[n] \\ u^\star[n] \end{bmatrix} \right) = 0. \quad (12)$$

Using the Schur Complement Lemma [2], we obtain that the nth optimal action is

$$u^\star[n] = (R + B^\mathsf{T} P_{n+1} B)^{-1} (S^\mathsf{T} + B^\mathsf{T} P_{n+1} A) x[n], \quad (13)$$

which corresponds to a linear control policy $u[n] = K_n x[n]$. The value function is $V_n(x[n]) = x[n]^\mathsf{T} P_n x[n]$ with

$$P_n = Q + A^\mathsf{T} P_{n+1} A - (S + A^\mathsf{T} P_{n+1} B)(R + B^\mathsf{T} P_{n+1} B)^{-1} (S^\mathsf{T} + B^\mathsf{T} P_{n+1} A). \quad (14)$$

The dynamic programming approach to the LQR Problem (6) consists of first computing $(P_0, \dots, P_{N-1})$ using the *difference Riccati equation* (Eq. 14) for $n = N-1, \dots, 0$ (i.e., backwards in time), starting from $P_N = Q_f$, then compute all the control actions $\mathbf{u}^\star = (u^\star[0], \dots, u^\star[N-1])$ using the *LQR control policy* (Eq. 13) for $n = 0, \dots, N-1$ (i.e., forward in time). This method, also known as *Riccati recursion* or *dynamic programming recursion*, is very efficient: it requires nothing but a sequence of simple matrix operations. Its complexity is dominated by the inverse $(R + B^\mathsf{T} P_{n+1} B)^{-1}$, which is cheap as the matrix is positive definite (that is, it admits a Cholesky decomposition).

In practice, modern solvers are able to solve the KKT System from the batch approach (Eq. 6) almost at the same complexity as required for the Riccati recursion. Therefore, which method to use almost boils down to preference (unless you don't have access to a good QP solver). However, the batch approach will remain applicable to constrained problems (which we discuss next week), whereas the Riccati recursion is mostly relevant to unconstrained LQ problems.

References

[1] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. *Model Predictive Control: Theory, Computation and Design*. 2nd ed. Nob Hill Publ., LLC., 2020. ISBN: 978-0-9759377-5-4.

[2] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. ISBN: 978-0-52-183378-3.