## 2.10 Model-Predictive Control (MPC)

### 2.10.1 From open-loop to feedback control

When we looked at the LQR controller (in section 2.5), we saw that we were trying to find a control input signal $u$ over a time-horizon, such that a cost function $J$ was minimized. Hence, for the discrete-time system case, we want to find a sequence of $N$ control inputs

$$\mathbf{u}(0), \mathbf{u}(1), \cdots, \mathbf{u}(N-1) \tag{2.99}$$

which will minimize the cost function

$$J = \sum_{k=0}^{N-1} \left[ \mathbf{x}^T(k)\mathbf{Q}\mathbf{x}(k) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k)) \right]. \tag{2.100}$$

where $N$ is called the horizon. Cost function (2.100) is minimized for systems represented by the discrete-time representation

$$\begin{cases} \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k), & \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) \end{cases} . \tag{2.101}$$

In the above formulation, summarized by (2.99), (2.100) and (2.101), we have, in essence, an *open-loop control law*. Indeed, given model (2.101) starting at initial state $x(0)$ (which we assume that we know or measure), the minimization of cost function/criterion (2.100) gives the sequence $\mathbf{u}(0), \mathbf{u}(1), \cdots, \mathbf{u}(N-1)$, which is then applied to system (2.101). It is open-loop because there is no feedback, ie the obtained state sequence $\mathbf{x}(1), \mathbf{x}(2), \cdots$ is not used to make corrections on $\mathbf{u}(0), \mathbf{u}(1), \cdots$!
A simple way to use the above strategy but add a feedback mechanism can be done as follows: start from $\mathbf{x}(0)$, obtain by cost minimization the input sequence $\mathbf{u}(0), \mathbf{u}(1), \cdots, \mathbf{u}(N-1)$, but apply *only* control input $\mathbf{u}(0)$ to system (2.101). Then measure the state at time $k = 1$, ie $\mathbf{x}(1)$. Move now the horizon of the cost minimization problem from $\{0, \cdots, N-1\}$ to $\{1, \cdots, N\}$, and, starting this time from $\mathbf{x}(1)$, obtain by cost minimization the input sequence $\mathbf{u}(1), \mathbf{u}(2), \cdots, \mathbf{u}(N)$, and apply control input $\mathbf{u}(1)$ to system (2.101). Proceed similarly for subsequent iterations.
This moving-horizon control strategy, which is based on feedback information from the state $\mathbf{x}$ at each iteration, is called *Receding-Horizon Control* or *Model-Predictive Control*[1].
Hence, mathematically, our cost function (2.100) can be slightly rewritten as

$$J(k) = \sum_{i=0}^{N-1} \left[ \mathbf{x}^T(k+i)\mathbf{Q}\mathbf{x}(k+i) + \mathbf{u}^T(k+i)\mathbf{R}\mathbf{u}(k+i)) \right], \tag{2.102}$$

for which we will try to find, at each iteration $k$, the control sequence

$$\mathbf{u}(k+0), \mathbf{u}(k+1), \cdots, \mathbf{u}(k+N-1) \tag{2.103}$$

[1]The term "predictive" comes from looking at future control inputs and future states at each iteration.

which will minimize (2.102), while we will only apply the first term $\mathbf{u}(k+0)$ of sequence (2.103) to system (2.101) at each iteration.
Note that, in addition to conventional LQR controllers, MPC also allow for the consideration of inequality constraints such as, for example,

$$\underline{\mathbf{u}} \leq \mathbf{u}(k) \leq \bar{\mathbf{u}}, \tag{2.104}$$

expression (2.104) being particularly useful to represent limits/saturations on actuators[2].

### 2.10.2 Quadratic Programming (QP)

Obtaining the sequence (2.103) that will minimize (2.102) under constraints such as (2.104) can generally *not* be computed explicitly. It can, however, be done numerically, using an optimization technique called *Quadratic Programming*, or QP for short.
Mathematically, quadratic programming consists in finding a vector $\mathbf{z}$ that will

$$\text{minimize } \frac{1}{2}\mathbf{z}^T\mathbf{H}\mathbf{z} + \mathbf{F}^T\mathbf{z}, \tag{2.105}$$

where $\mathbf{F}$ is a vector such that $\dim(\mathbf{z}) = \dim(\mathbf{F})$ and square matrix $\mathbf{H}$ is such that term $\mathbf{z}^T\mathbf{H}\mathbf{z}$ is a scalar.
To (2.105) can be adjoined constraints of several types, ie equality constraints of the type $\mathbf{G}\mathbf{z} = \mathbf{q}$ and inequality constraints such as $\mathbf{W}\mathbf{z} \leq \mathbf{v}$ or $\underline{\mathbf{z}} \leq \mathbf{z} \leq \bar{\mathbf{z}}$.
In Matlab, quadratic programming can simply be done with the command `quadprog`, while many options are also available in Python (see for example the package `cvxopt`).

### 2.10.3 From QP to linear MPC

If one wants to use the numerical tools available for QP (using Matlab or Python, for example), the MPC formulation summarized by expressions (2.102), (2.103), (2.104) should be put under the form (2.105) with the corresponding constraints. To do that, let us consider the simpler particular case where $N = 3$, and where, *at each iteration* $k$, we have[3]

$$\begin{aligned} J &= \sum_{i=0}^{N-1} \left[\mathbf{x}^T(i)\mathbf{Q}\mathbf{x}(i) + \mathbf{u}^T(i)\mathbf{R}\mathbf{u}(i))\right] \\ &= \mathbf{x}^T(0)\mathbf{Q}\mathbf{x}(0) + \mathbf{x}^T(1)\mathbf{Q}\mathbf{x}(1) + \mathbf{x}^T(2)\mathbf{Q}\mathbf{x}(2) \\ &\quad + \mathbf{u}^T(0)\mathbf{R}\mathbf{u}(0) + \mathbf{u}^T(1)\mathbf{R}\mathbf{x}(1) + \mathbf{u}^T(2)\mathbf{R}\mathbf{u}(2) \\ &= \begin{bmatrix}\mathbf{x}(0)\\ \mathbf{x}(1)\\ \mathbf{x}(2)\end{bmatrix}^T \begin{bmatrix}\mathbf{Q} & & \\ & \mathbf{Q} & \\ & & \mathbf{Q}\end{bmatrix} \begin{bmatrix}\mathbf{x}(0)\\ \mathbf{x}(1)\\ \mathbf{x}(2)\end{bmatrix} + \begin{bmatrix}\mathbf{u}(0)\\ \mathbf{u}(1)\\ \mathbf{u}(2)\end{bmatrix}^T \begin{bmatrix}\mathbf{R} & & \\ & \mathbf{R} & \\ & & \mathbf{R}\end{bmatrix} \begin{bmatrix}\mathbf{u}(0)\\ \mathbf{u}(1)\\ \mathbf{u}(2)\end{bmatrix} \end{aligned}$$

[2]Expression (2.104) should be understood in a term-by-term sense, ie the $i$-th term of control input vector $\mathbf{u}(k)$ is bounded from above and below, ie $\underline{u}_i \leq u_i(k) \leq \bar{u}_i$

[3]For simplicity and better readibility, we drop the dependency on $k$ in the following mathematical development, knowing that the subsequent calculations are valud for each iteration $k$.

so that we get the more compact expression

$$J = \mathbf{X}^T \mathbb{Q} \mathbf{X} + U^T \mathbb{R} \mathbf{U}, \tag{2.106}$$

with $\mathbf{X} = [\mathbf{x}^T(0), \mathbf{x}^T(1), \cdots, \mathbf{x}^T(N-1)]^T$, with the matrices

$$\mathbb{Q} = \begin{bmatrix} \mathbf{Q} & & \\ & \mathbf{Q} & \\ & & \mathbf{Q} \end{bmatrix} \quad \text{and} \quad \mathbb{R} = \begin{bmatrix} \mathbf{R} & & \\ & \mathbf{R} & \\ & & \mathbf{R} \end{bmatrix}, \tag{2.107}$$

and where $\mathbf{U}$ is the vector containing the sequence of control inputs we want to find, ie

$$\mathbf{U} = [\mathbf{u}^T(0), \mathbf{u}^T(1), \cdots, \mathbf{u}^T(N-1)]^T. \tag{2.108}$$

Then, note that state-space representation (2.101) means that

$$\begin{aligned} \mathbf{x}(0) &= \mathbf{x}(0) \\ \mathbf{x}(1) &= \mathbf{A}\mathbf{x}(0) + \mathbf{B}\mathbf{u}(0) \\ \mathbf{x}(2) &= \mathbf{A}\mathbf{x}(1) + \mathbf{B}\mathbf{u}(1) \\ &= \mathbf{A}\left[\mathbf{A}\mathbf{x}(0) + \mathbf{B}\mathbf{u}(0)\right] + \mathbf{B}\mathbf{u}(1) \\ &= \mathbf{A}^2\mathbf{x}(0) + \mathbf{A}\mathbf{B}\mathbf{u}(0) + \mathbf{B}\mathbf{u}(1). \end{aligned}$$

Rewriting the above in vectorial form, we get

$$\begin{bmatrix} \mathbf{x}(0) \\ \mathbf{x}(1) \\ \mathbf{x}(2) \end{bmatrix} = \begin{bmatrix} I \\ \mathbf{A} \\ \mathbf{A}^2 \end{bmatrix} \mathbf{x}(0) + \begin{bmatrix} 0 & 0 & 0 \\ \mathbf{B} & 0 & 0 \\ \mathbf{A}\mathbf{B} & \mathbf{B} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}(0) \\ \mathbf{u}(1) \\ \mathbf{u}(2) \end{bmatrix} \tag{2.109}$$

so that we have

$$\mathbf{X} = \mathbb{A}\mathbf{x}(0) + \mathbb{B}\mathbf{U} \tag{2.110}$$

with matrices

$$\mathbb{A} = \begin{bmatrix} I \\ \mathbf{A} \\ \mathbf{A}^2 \end{bmatrix} \quad \text{and} \quad \mathbb{B} = \begin{bmatrix} 0 & 0 & 0 \\ \mathbf{B} & 0 & 0 \\ \mathbf{A}\mathbf{B} & \mathbf{B} & 0 \end{bmatrix}. \tag{2.111}$$

Putting then (2.110) into (2.106), we have

$$\begin{aligned} J &= \left[\mathbb{A}\mathbf{x}(0) + \mathbb{B}\mathbf{U}\right]^T \mathbb{Q} \left[\mathbb{A}\mathbf{x}(0) + \mathbb{B}\mathbf{U}\right] + \mathbf{U}^T \mathbb{R} \mathbf{U} \\ &= \left[\mathbf{x}^T(0)\mathbb{A}^T + \mathbf{U}^T\mathbb{B}^T\right] \mathbb{Q} \left[\mathbb{A}\mathbf{x}(0) + \mathbb{B}\mathbf{U}\right] + \mathbf{U}^T \mathbb{R} \mathbf{U} \\ &= \mathbf{x}^T(0)\mathbb{A}^T\mathbb{Q}\mathbb{A}\mathbf{x}(0) + \mathbf{x}^T(0)\mathbb{A}^T\mathbb{Q}\mathbb{B}\mathbf{U} + \mathbf{U}^T\mathbb{B}^T\mathbb{Q}\mathbb{A}\mathbf{x}(0) \\ &\qquad + \mathbf{U}^T\mathbb{B}^T\mathbb{Q}\mathbb{B}\mathbf{U} + \mathbf{U}^T\mathbb{R}\mathbf{U} \end{aligned}$$

The last equality consisting of five (scalar) terms, combine, respectively, the second and third terms together, and the fourth and fifth terms together, to get

$$J = \mathbf{x}^T(0)\mathbb{A}^T\mathbb{Q}\mathbb{A}\mathbf{x}(0) + 2\mathbf{x}^T(0)\mathbb{A}^T\mathbb{Q}\mathbb{B}\mathbf{U} + \mathbf{U}^T\left[\mathbb{B}^T\mathbb{Q}\mathbb{B} + \mathbb{R}\right]\mathbf{U}. \tag{2.112}$$

Cost function (2.113) should be minimized with respect to vector $\mathbf{U}$. Since term $\mathbf{x}^T(0)\mathbb{A}^T\mathbb{Q}\mathbb{A}\mathbf{x}(0)$ is known, then finding $\mathbf{U}$ that minimizes $J$ is the same as finding $\mathbf{U}$ that minimizes

$$J' = \mathbf{U}^T\left[\mathbb{B}^T\mathbb{Q}\mathbb{B} + \mathbb{R}\right]\mathbf{U} + 2\mathbf{x}^T(0)\mathbb{A}^T\mathbb{Q}\mathbb{B}\mathbf{U}, \tag{2.113}$$

which has the same form as quadratic programming problem (2.105), with $\mathbf{z} = \mathbf{U}$,

$$\mathbf{H} = 2\left[\mathbb{B}^T\mathbb{Q}\mathbb{B} + \mathbb{R}\right] \tag{2.114}$$

and

$$\mathbf{F} = 2\left[\mathbf{x}^T(0)\mathbb{A}^T\mathbb{Q}\mathbb{B}\right]^T. \tag{2.115}$$