Model Predictive Control

Adam Heins

June 27, 2024

Here we derive some basic model predictive controllers.

1 Linear MPC

Suppose we have a discrete-time linear time-invariant system of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k,$$

where $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$ are the state and input at the kth timestep, respectively. Consider the optimal control problem

$$\begin{aligned} \text{minimize} \quad & \ell_f(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} \ell(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ \text{subject to} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \quad k = 0, \dots, N, \\ & \boldsymbol{u}_k \in \mathcal{U}, \quad k = 0, \dots, N-1, \\ & \boldsymbol{x}_k \in \mathcal{X}, \quad k = 0, \dots, N-1, \\ & \boldsymbol{x}_N \in \mathcal{X}_f, \end{aligned} \tag{1}$$

where we are trying to find the control inputs $\boldsymbol{u}$ that minimize the total cost (running cost ℓ and terminal cost ℓ_f) over a horizon of N steps subject to the dynamics and state and input constraints.

Let us consider how to solve (1). Define the *lifted* state and input vectors

$$\bar{\boldsymbol{x}} = \left[\boldsymbol{x}_0^T, \dots, \boldsymbol{x}_N^T\right]^T, \qquad \bar{\boldsymbol{u}} = \left[\boldsymbol{u}_0^T, \dots, \boldsymbol{u}_{N-1}^T\right]^T,$$

and let us take the costs to be

$$\begin{aligned} \ell(\boldsymbol{x}, \boldsymbol{u}) &= \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}, \\ \ell_f(\boldsymbol{x}) &= \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}, \end{aligned}$$

for positive definite $\boldsymbol{Q}$ and $\boldsymbol{R}$. We have assumed that the terminal cost is the same as the running cost for simplicity. We will take the input constraints to be the bounds $\mathcal{U} = \{\boldsymbol{u} \mid \boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max}\}$ and we will omit the state constraints entirely. So far we have assumed that we are regulating the system to the origin $\boldsymbol{x} = \boldsymbol{0}$, but we will also discuss tracking a desired trajectory $\bar{\boldsymbol{x}}_d$.

1.1 Multiple shooting

In a multiple shooting approach, we optimize over both $\bar{\boldsymbol{x}}$ and $\bar{\boldsymbol{u}}$. Let us define the matrices

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{I} & \boldsymbol{0} & \\ \boldsymbol{0} & \boldsymbol{A} & -\boldsymbol{I} & \\ & & \ddots & \ddots \end{bmatrix} \qquad \bar{\boldsymbol{B}} = \mathrm{diag}(\boldsymbol{B})^N$$
$$\bar{\boldsymbol{Q}} = \mathrm{diag}(\boldsymbol{Q})^{N+1}, \qquad \bar{\boldsymbol{R}} = \mathrm{diag}(\boldsymbol{R})^N,$$

where $\boldsymbol{I}$ is the identity matrix and $\mathrm{diag}(\boldsymbol{X})^n$ is a block diagonal matrix with n blocks of the matrix $\boldsymbol{X}$. Given the actual value of the current state $\hat{\boldsymbol{x}}_0$, the OCP (1) can be formulated as the quadratic program

$$\begin{aligned} \underset{\bar{\boldsymbol{x}},\bar{\boldsymbol{u}}}{\text{minimize}} \quad & \bar{\boldsymbol{x}}^T\bar{\boldsymbol{Q}}\bar{\boldsymbol{x}} + \bar{\boldsymbol{u}}^T\bar{\boldsymbol{R}}\bar{\boldsymbol{u}} \\ \text{subject to} \quad & \boldsymbol{x}_0 = \hat{\boldsymbol{x}}_0 \\ & \bar{\boldsymbol{A}}\bar{\boldsymbol{x}} + \bar{\boldsymbol{B}}\bar{\boldsymbol{u}} = \boldsymbol{0} \\ & \bar{\boldsymbol{u}}_{\min} \le \bar{\boldsymbol{u}} \le \bar{\boldsymbol{u}}_{\max}. \end{aligned} \tag{2}$$

If we want to track a desired trajectory $\bar{\boldsymbol{x}}_d$, then we add the linear term $-2\bar{\boldsymbol{x}}_d^T\bar{\boldsymbol{Q}}\bar{\boldsymbol{x}}$ to the cost.

1.2 Single shooting

In a single shooting approach, we eliminate the state variables and optimize only over $\bar{\boldsymbol{u}}$. Let us (re)define the matrices

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix}, \qquad \bar{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots \\ \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} & \cdots \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \boldsymbol{0} & \\ \boldsymbol{A}^2\boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \\ \vdots & & & \ddots \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & & & \end{bmatrix},$$

such that $\bar{\boldsymbol{x}} = \bar{\boldsymbol{A}}\hat{\boldsymbol{x}}_0 + \bar{\boldsymbol{B}}\bar{\boldsymbol{u}}$. The objective function becomes

$$\begin{aligned} \bar{\boldsymbol{x}}^T\bar{\boldsymbol{Q}}\bar{\boldsymbol{x}} + \bar{\boldsymbol{u}}^T\bar{\boldsymbol{R}}\bar{\boldsymbol{u}} &= (\bar{\boldsymbol{A}}\hat{\boldsymbol{x}}_0 + \bar{\boldsymbol{B}}\bar{\boldsymbol{u}})^T\bar{\boldsymbol{Q}}(\bar{\boldsymbol{A}}\hat{\boldsymbol{x}}_0 + \bar{\boldsymbol{B}}\bar{\boldsymbol{u}}) + \bar{\boldsymbol{u}}^T\bar{\boldsymbol{R}}\bar{\boldsymbol{u}} \\ &= \hat{\boldsymbol{x}}_0^T\bar{\boldsymbol{A}}^T\bar{\boldsymbol{Q}}\bar{\boldsymbol{A}}\hat{\boldsymbol{x}}_0 + 2\hat{\boldsymbol{x}}_0^T\bar{\boldsymbol{A}}^T\bar{\boldsymbol{Q}}\bar{\boldsymbol{B}}\bar{\boldsymbol{u}} + \bar{\boldsymbol{u}}^T(\bar{\boldsymbol{R}} + \bar{\boldsymbol{B}}^T\bar{\boldsymbol{Q}}\bar{\boldsymbol{B}})\bar{\boldsymbol{u}}, \end{aligned}$$

where the first term is constant and so can be omitted from the optimization problem. We have also eliminated the dynamics equality constraints, so (1) just becomes the quadratic program

$$\begin{aligned} \underset{\bar{\boldsymbol{u}}}{\text{minimize}} \quad & (1/2)\bar{\boldsymbol{u}}^T(\bar{\boldsymbol{R}} + \bar{\boldsymbol{B}}^T\bar{\boldsymbol{Q}}\bar{\boldsymbol{B}})\bar{\boldsymbol{u}} + \hat{\boldsymbol{x}}_0^T\bar{\boldsymbol{A}}^T\bar{\boldsymbol{Q}}\bar{\boldsymbol{B}}\bar{\boldsymbol{u}} \\ \text{subject to} \quad & \bar{\boldsymbol{u}}_{\min} \le \bar{\boldsymbol{u}} \le \bar{\boldsymbol{u}}_{\max}. \end{aligned} \tag{3}$$

If we want to track a desired trajectory $\bar{\boldsymbol{x}}_d$, then we modify the linear term of the cost to $(\bar{\boldsymbol{A}}\hat{\boldsymbol{x}}_0 - \bar{\boldsymbol{x}}_d)^T\bar{\boldsymbol{Q}}\bar{\boldsymbol{B}}\bar{\boldsymbol{u}}$.

2 Nonlinear MPC

Suppose now we have a discrete-time nonlinear system of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k). \tag{4}$$

We will proceed in a similar way as the linear MPC case by *linearizing* the system about a trajectory $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$, yielding

$$\boldsymbol{A}_k = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{(\tilde{\boldsymbol{x}}_k, \tilde{\boldsymbol{u}}_k)}, \qquad \boldsymbol{B}_k = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{(\tilde{\boldsymbol{x}}_k, \tilde{\boldsymbol{u}}_k)}.$$

At each timestep, our approach will be to iteratively linearize about some nominal trajectory, then formulate and solve a QP similar to the above. However, differently from the linear case, we will iteratively re-linearize and solve the QP, until the solution has converged (or we have run out of compute time). That means that each QP will solve for an update to the current solution, rather than the solution directly. We will use a multiple shooting approach.

Let $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{u}}$ be the lifted state and input trajectories from the previous iteration (we can initialize $\tilde{\boldsymbol{u}}$ from the previous timestep or simply with all zeros, then roll out $\tilde{\boldsymbol{x}}$ using the nonlinear dynamics (4)). Then we want to solve for updates $\delta\bar{\boldsymbol{x}}$ and $\delta\bar{\boldsymbol{u}}$ to obtain a new optimal trajectory

$$\begin{aligned} \bar{\boldsymbol{x}} &= \tilde{\boldsymbol{x}} + \delta\bar{\boldsymbol{x}}, \\ \bar{\boldsymbol{u}} &= \tilde{\boldsymbol{u}} + \delta\bar{\boldsymbol{u}}. \end{aligned}$$

Let us define the matrices

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_0 & -\boldsymbol{I} & \boldsymbol{0} & & \\ \boldsymbol{0} & \boldsymbol{A}_1 & -\boldsymbol{I} & & \\ & & \ddots & \ddots & \\ & & & \boldsymbol{A}_{N-1} & -\boldsymbol{I} \end{bmatrix}, \quad \bar{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{B}_0 & \boldsymbol{0} & & \\ \boldsymbol{0} & \boldsymbol{B}_1 & & \\ & & \ddots & \\ & & & \boldsymbol{B}_{N-1} \end{bmatrix},$$

Substituting into (2), we get

$$\begin{aligned} \underset{\delta\bar{\boldsymbol{x}}, \delta\bar{\boldsymbol{u}}}{\text{minimize}} \quad & (1/2)\delta\bar{\boldsymbol{x}}^T\bar{\boldsymbol{Q}}\delta\bar{\boldsymbol{x}} + (1/2)\delta\bar{\boldsymbol{u}}^T\bar{\boldsymbol{R}}\delta\bar{\boldsymbol{u}} + \tilde{\boldsymbol{x}}^T\bar{\boldsymbol{Q}}\delta\bar{\boldsymbol{x}} + \tilde{\boldsymbol{u}}^T\bar{\boldsymbol{R}}\delta\bar{\boldsymbol{u}} \\ \text{subject to} \quad & \delta\boldsymbol{x}_0 = \boldsymbol{0} \\ & \bar{\boldsymbol{A}}\delta\bar{\boldsymbol{x}} + \bar{\boldsymbol{B}}\delta\bar{\boldsymbol{u}} + \bar{\boldsymbol{A}}\tilde{\boldsymbol{x}} + \bar{\boldsymbol{B}}\tilde{\boldsymbol{u}} = \boldsymbol{0} \\ & \bar{\boldsymbol{u}}_{\min} - \tilde{\boldsymbol{u}} \leq \delta\bar{\boldsymbol{u}} \leq \bar{\boldsymbol{u}}_{\max} - \tilde{\boldsymbol{u}}, \end{aligned} \tag{5}$$

which we can iterate at each timestep as desired. If we want to track a desired trajectory $\bar{\boldsymbol{x}}_d$, then we add the linear term $-\bar{\boldsymbol{x}}_d^T\bar{\boldsymbol{Q}}\delta\bar{\boldsymbol{x}}$ to the cost. Note that if more complex constraints are included in the OCP, then we likely need to employ a filter or line-search method to determine the optimal step size, rather than taking a full step at each iteration.