Lab 8B: Closed Loop Control

Ignacio Hounie, Damian Owerko, and Alejandro Ribeiro*

November 4, 2024

1 Model Predictive Control (MPC)

In Lab 5A we tried to control a dynamical system with model based optimization. This strategy failed because of the mismatch between the model and the true system.

We illustrate this issue in Figure 1. In model based optimization we find control actions $\hat{\mathbf{u}}(t)$ that are optimal for the model of the true dynamical system. This is represented by the bottom part of the diagram where we show that the controller is optimizing with respect to model trajectories $\hat{\mathbf{x}}(t)$. Once we find actions that are optimal for the model, we turn around and implement them in the true model. This is shown in the top part of the diagram where the control input of the true dynamical system is $\mathbf{u}(t) = \hat{\mathbf{u}}^*(t)$.

If we initialize the model and the true system with the same initial conditions, the trajectory $\hat{\mathbf{x}}(t)$ of the model and the trajectory $\mathbf{x}(t)$ of the true system are initially close. The first few control actions $\hat{\mathbf{u}}^*(t)$ are therefore good control actions for the true system. Eventually, however, the trajectory $\hat{\mathbf{x}}(t)$ of the model and the trajectory $\mathbf{x}(t)$ of the true system deviate. At this point actions $\hat{\mathbf{u}}^*(t)$ that are chosen because they are optimal for the model are far from good for the true system.

*In alphabetical order.

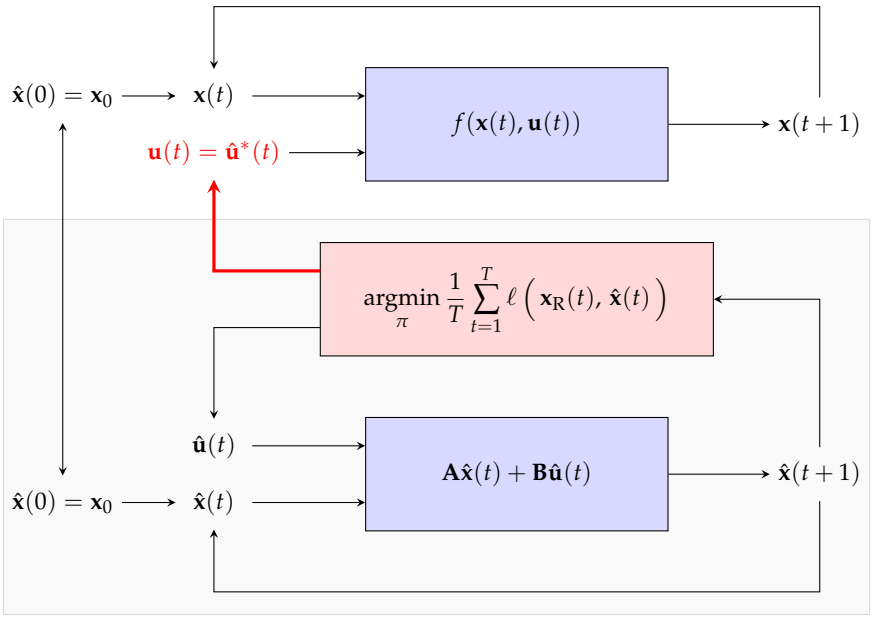

Figure 1. Mismatch in Model Based Optimization. The mismatch between the true dynamical system (top) and its model (bottom) accumulates over time. Eventually, the states $\hat{\mathbf{x}}(t)$ and $\mathbf{x}(t)$ of the model and the true system deviate. When this happens, actions $\hat{\mathbf{u}}^*(t)$ that are optimal for the model trajectory are not good for the true trajectory of the dynamical system.

This explanation suggests a solution. What we need to do is reinitialize the model every so often. This is called model predictive control (MPC) and is illustrated in Figure 2.

In MPC we define a window of length W and solve the optimization problem,

$$\begin{aligned} \hat{\mathbf{u}}^*(t : t+W) \;=\; & \underset{\hat{\mathbf{u}}(s)}{\text{argmin}} \; \frac{1}{W} \sum_{s=t+1}^{t+W} \ell\left(\mathbf{x}_R(s), \hat{\mathbf{x}}(s)\right), \\ & \text{with} \quad \hat{\mathbf{x}}(s+1) = \mathbf{A}\hat{\mathbf{x}}(s) + \mathbf{B}\hat{\mathbf{u}}(s), \\ & \hat{\mathbf{x}}(t) = \mathbf{x}(t). \end{aligned} \tag{1}$$

This is the same model based optimization problem we encountered in

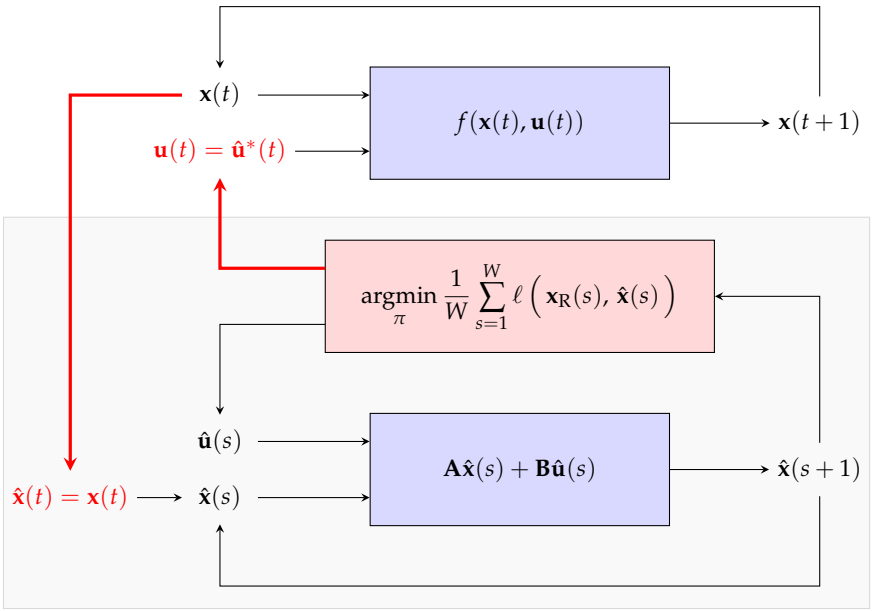

Figure 2. Model Predictive Control. The bottom part of the diagram uses the model to determine control actions $\mathbf{u}^*(t:t+W)$ that are optimal for driving *model* predicted trajectories $\hat{\mathbf{x}}(s)$ [cf. (1)]. We implement the control action $\mathbf{u}(t) = \hat{\mathbf{u}}^*(t)$ in the *true* system (top). We preclude the accumulation of errors by reinitializing the model at each time step t (left). We say that we close the control loop.

Lab 5A, except for two modifications: (i) The initial condition is $\hat{\mathbf{x}}(t) = \mathbf{x}(t)$. (ii) The merit of the trajectory is evaluated between times $s = t+1$ and $s = t+W$. These are not significant differences. Modification (ii) is just a practical consideration to reduce the complexity of the optimization problem. This is achieved by reducing the time horizon from T to W. Modification (i) is just a shift in time. We initialize the model with the state at time t instead of initializing the model with the state at time 0

The significant difference is that we now extract the control action $\mathbf{u}^*(t)$ and execute this action in the true dynamical system. The remaining actions $\mathbf{u}^*(t+1:t+W)$ that are also part of the solution of (1) are discarded. We then proceed to advance time to $t+1$ and solve the model based optimization in (1) with initial condition $\hat{\mathbf{x}}(t+1) = \mathbf{x}(t+1)$.

The MPC controller in Figure 2 is what we call a *closed* loop control sys-

tem. We observe the state $\mathbf{x}(t)$ at time t and use this information to select the control action $\mathbf{u}(t) = \hat{\mathbf{u}}^*(t)$. The effect of this control action is to move the true system into state $\mathbf{x}(t+1)$. The terminology of closing the loop refers to the fact that we now observe the state $\mathbf{x}(t+1)$ and use this observation to select the control action $\mathbf{u}(t+1) = \hat{\mathbf{u}}^*(t+1)$. This is different from the *open* loop control strategy in Figure 1 in which we observe the initial condition $\mathbf{x}(0) = \mathbf{x}_0$ but never again observe the state of the true system.

Task 1 Leverage the model based optimization code of Lab 5A to run MPC control on the car simulator. Implementing MPC requires that you implement the following steps at all times t,

- **(S1)** Observe the current state $\mathbf{x}(t)$ of the true system.
- **(S2)** Initialize the model system with $\hat{\mathbf{x}}(t) = \mathbf{x}(t)$.
- **(S3)** Solve (1). The outcome is a sequence $\mathbf{u}^*(t : t+W)$.
- **(S4)** Implement control action $\mathbf{u}(t) = \hat{\mathbf{u}}^*(t)$ in the true system.

Step (S3) is where we compute the control action $\mathbf{u}(t) = \hat{\mathbf{u}}^*(t)$. We actually compute a sequence of control actions $\mathbf{u}^*(t : t+W)$ but we discard all entries of the sequence but the first. Steps (S4) implements the control action in the true system. Steps (S1) and (S2) close the control loop by observing the current state of the true system and feeding it as an input to the model system.

Plot the reference trajectory $\mathbf{x}_{\mathrm{R}}(t)$ and the trajectory $\mathbf{x}(t)$ generated by implementation of the MPC controller. Slow down the plot to create a movie of the controlled car following the reference trajectory.

Evaluate and plot the loss between the generated trajectory and the reference trajectory. ■

The most important conclusion of Task 1 is summarized in Figure 3. The trajectory generated by the MPC controller follows the reference trajectory with small loss. This is because closing the loop precludes the accumulation of errors.

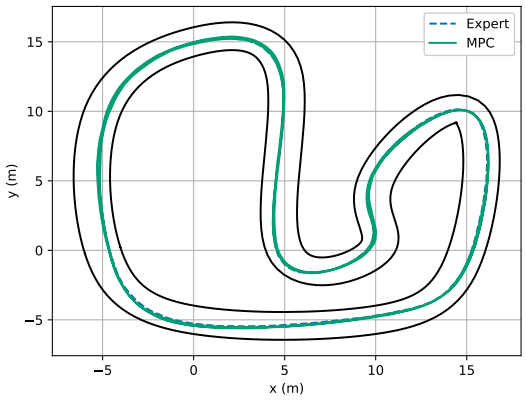

Figure 3. Trajectory Generated by the Model Predictive Controller. The MPC controller generates a trajectory that follows the reference trajectory with small error. This is because closing the loop precludes the accumulation of errors.

It is interesting to think of the model part of the controller as a simulation of the true dynamical system. We use this simulation to evaluate the merit of different control actions as a way of deciding which control action to execute. This is a very human way of thinking. We use models to simulate the effects of different courses of action. We then execute the action that our simulations deem best. Because simulations and reality are in the end different, we observe the outcome of our actions and repeat the simulation to select a new course of action.

2 Imitation Learning

The MPC controller in (1) has a high computation cost as it requires the solution of an optimization problem in each time step. This is a challenge because the time between time steps can be short. In the car driving scenario we are considering, the sampling time is $T_s = 50ms$.

We can avoid this computation cost using learning. One approach is to consider the solutions of (1) and train a learning parametrization that maps the initialization $\hat{\mathbf{x}}(t) = \mathbf{x}(t)$ into the control action $\mathbf{u}(t) = \hat{\mathbf{u}}^*(t)$. We

do that by introducing a learning parametrization $\pi(\mathbf{x}(t);\theta)$ and defining the optimization problem,

$$\theta^* \;=\; \underset{\theta}{\text{argmin}} \;\frac{1}{T}\sum_{t=1}^{T} \ell\,\Big(\,\hat{\mathbf{u}}^*(t),\, \pi\big(\mathbf{x}(t);\theta\big)\,\Big). \tag{2}$$

In this problem $\mathbf{x}(t)$ is the initialization of the MPC controller in (1) and $\hat{\mathbf{u}}^*(t)$ is the optimal control action produced as the corresponding solution of the optimization problem.

We say that (2) is an imitation problem because we are training a policy $\pi(\mathbf{x}(t);\theta)$ that imitates the actions of the MPC controller. The advantage of training this imitation policy is that we can train it offline, as we always do. At execution time we just evaluate the policy $\pi(\mathbf{x}(t);\theta^*)$ and execute the corresponding control action. The computation cost of this evaluation is smaller than the cost of solving (2).

Notice that in (2) we assume that the states $\mathbf{x}(t)$ are states in the trajectory of the dynamical system generated by (1). This does not have to be the case. We can feed any set of input states. For instance, in the circuit navigation problem we are given a number of expert trajectories. We can therefore consider the observed states $\mathbf{x}_q(t)$ and replace (2) with

$$\theta^* \;=\; \underset{\theta}{\text{argmin}} \;\frac{1}{Q}\sum_{q=1}^{Q}\;\frac{1}{T}\sum_{t=1}^{T} \ell\,\Big(\,\hat{\mathbf{u}}_q^*(t),\, \pi\big(\mathbf{x}_q(t);\theta\big)\,\Big), \tag{3}$$

in which $\hat{\mathbf{u}}_q^*(t)$ is the solution of the MPC control optimization problem in (1) when the initial condition is $\hat{\mathbf{x}}(t) = \mathbf{x}_q(t)$.

A third possibility is to dither the expert trajectories in (3). That is, for each state $\mathbf{x}_q(t)$ we generate D states $\tilde{\mathbf{x}}_{qd}(t)$ by adding a random perturbation $\mathbf{w}_{qd}(t)$,

$$\tilde{\mathbf{x}}_{qd}(t) = \mathbf{x}_q(t) + \mathbf{w}_{qd}(t). \tag{4}$$

We then reformulate (3) by considering all trajectories, all points in time, and all dithered states. This leads to the optimization problem,

$$\theta^* \;=\; \underset{\theta}{\text{argmin}} \;\frac{1}{Q}\sum_{q=1}^{Q}\;\frac{1}{T}\sum_{t=1}^{T}\;\frac{1}{D}\sum_{d=1}^{D} \ell\,\Big(\,\hat{\mathbf{u}}_q^*(t),\, \pi\big(\tilde{\mathbf{x}}_{qd}(t);\theta\big)\,\Big), \tag{5}$$

in which $\hat{\mathbf{u}}_q^*(t)$ is reinterpreted as the solution of the MPC control optimization problem in (1) when the initial condition is $\mathbf{x}(t) = \tilde{\mathbf{x}}_{qd}(t)$. Adding dithering to the training set improves the robustness of the learned controller.

The optimization problems in (2), (3), and (5) have the same objective but take averages over different datasets. Thus, their relative merits depend on which of the two set of input states is more representative of the states that are to be expected at execution time. The possible mismatch between the states that we choose for optimizing θ and the states that are actually seen at execution time is a significant challenge when learning to imitate control policies of dynamical systems.

Task 2 Learn a policy that imitates the MPC controller. You can choose any parametrization you like, but to keep matters simple we recommend that you use a linear parameterization,

$$\pi\big(\mathbf{x}(t);\theta\big) = \mathbf{\Theta}(\mathbf{x}(t) - \mathbf{x}_{\mathrm{R}}(t)) \tag{6}$$

You can also use any loss you want. If you use a linear parameterization, a quadratic loss is a good idea.Train the imitation policy by solving (5). Consider $D = 1$ dithered state per $\mathbf{x}_{qd}(t)$ and use white Gaussian noise $\mathbf{w}_{qd}(t)$ with mean $\mathbb{E}[\mathbf{w}_{qd}(t)] = 0$ and variance $\mathbb{E}[\mathbf{w}_{qd}^2(t)] = 0.09$.

Evaluate the control policy $\pi(\mathbf{x}(t);\theta^*)$ on the car simulator. Plot the reference trajectory $\mathbf{x}_R(t)$ and the trajectory $\mathbf{x}(t)$ generated by the imitation policy. Slow down the plot to create a movie of the controlled car following the reference trajectory.

Evaluate and plot the loss between the generated trajectory and the reference trajectory. ■

3 Model Predictive Control Learning

A second alternative to learn a control policy is direct incorporation of the parameterization in (1). To do that, we consider a fixed parameter θ and the policy $\pi(\mathbf{x};\theta)$ that is generated by this parameter. If we execute

this policy on the system's *model* we incur the cost

$$\ell_{\text{MPC}}\big(\mathbf{x}(t);\theta\big) \;=\; \frac{1}{W}\sum_{s=t+1}^{t+W} \ell\Big(\mathbf{x}_{\text{R}}(s),\,\hat{\mathbf{x}}(s)\Big),$$

$$\text{with} \qquad \hat{\mathbf{x}}(s+1) = \mathbf{A}\hat{\mathbf{x}}(s) + \mathbf{B}\hat{\mathbf{u}}(s),$$

$$\hat{\mathbf{x}}(t) = \mathbf{x}(t),$$

$$\hat{\mathbf{u}}(t) = \pi\big(\mathbf{x}(s);\theta\big). \tag{7}$$

As is has been true throughout this chapter, this is not the true cost incurred by the policy $\pi(\mathbf{x};\theta)$ in the actual system. To evaluate this cost we would need to propagate actions through the system function $f(x(t),u(t))$. This is impossible because we do not know this function. The first merit of (7) is that it can be evaluated. Its second merit is that if model and reality are not that different, a policy that works well in the model works well in the true system.

The loss function defined in (7) is different from loss functions that we have encountered before because it evaluates the merit of the policy $\pi(\mathbf{x};\theta)$ over a trajectory of W time units. A strange loss function it may be, but a loss function nonetheless. We can then use it to propose the learning problem,

$$\theta^* \;=\; \underset{\theta}{\text{argmin}}\,\frac{1}{T}\sum_{t=1}^{T}\ell_{\text{MPC}}\big(\mathbf{x}(t);\theta\big). \tag{8}$$

As in Equation (2) of Section 2 states $\mathbf{x}(t)$ in (8) are drawn from the trajectory generated by the MPC controller. As is also the case of Section 2 this does not have to be the case. We can train by minimizing the loss in (7) over any set of states $\mathbf{x}(t)$. In particular, if we use expert trajectories to define the optimal parameter θ^* we replace (8) with

$$\theta^* \;=\; \underset{\theta}{\text{argmin}}\;\frac{1}{Q}\sum_{q=1}^{Q}\;\frac{1}{T}\sum_{t=1}^{T}\ell_{\text{MPC}}\big(\mathbf{x}_q(t);\theta\big). \tag{9}$$

Observe that This formulation is akin to (3). Except that instead of searching for a parameter θ that imitates the MPC policy $\hat{\mathbf{u}}_q^*(t)$, we search for a parameter θ that minimizes the MPC cost in (7).

As the reader may be expecting, the use of dithered trajectories [cf. (4)] is

also justified. In such case we would replace (9) by

$$\theta^* \;=\; \underset{\theta}{\text{argmin}} \;\frac{1}{Q}\sum_{q=1}^{Q}\;\frac{1}{T}\sum_{t=1}^{T}\;\frac{1}{D}\sum_{d=1}^{D}\ell_{\text{MPC}}\big(\mathbf{x}_{qd}(t);\theta\big). \tag{10}$$

This formulation is akin to (5) but we seek to minimize the MPC cost in (7) instead of imitating the MPC policy $\hat{\mathbf{u}}_q^*(t)$. As in imitation learning, adding dithering to the training set improves the robustness of the learned controller.

Task 3 Learn a policy that minimizes the MPC loss defined in (7). You can choose any parametrization you like, but to keep matters simple we recommend that you use a linear parameterization as in (6). You can train the imitation policy by solving either (8), (9) or (10). The three choices have merit. Whichever you use, explain your reasons for choosing it.

Evaluate the control policy $\pi(\mathbf{x}(t);\theta^*)$ on the car simulator. Plot the reference trajectory $\mathbf{x}_R(t)$ and the trajectory $\mathbf{x}(t)$ generated by the learned policy. Slow down the plot to create a movie of the controlled car following the reference trajectory.

Evaluate and plot the loss between the generated trajectory and the reference trajectory. ■

4 Report

Do not take much time to prepare a lab report. We do not want you to report your code and we don't want you to report your work. Just give us answers to questions we ask. Specifically give us the following:

Question	**Report deliverable**
Task 1	Plot of reference trajectory $\mathbf{x}_{\mathrm{R}}(t)$ and controlled trajectory $\mathbf{x}(t)$.
	Plot of loss associated with controlled trajectory $\mathbf{x}(t)$ relative to reference trajectory $\mathbf{x}_{\mathrm{R}}(t)$.
Task 2	Plot of reference trajectory $\mathbf{x}_{\mathrm{R}}(t)$ and controlled trajectory $\mathbf{x}(t)$.
	Plot of loss associated with controlled trajectory $\mathbf{x}(t)$ relative to reference trajectory $\mathbf{x}_{\mathrm{R}}(t)$.
	Choice of parameterization and loss.
	Choice of training set. Explain choice in one paragraph.
Task 3	Plot of reference trajectory $\mathbf{x}_{\mathrm{R}}(t)$ and controlled trajectory $\mathbf{x}(t)$.
	Plot of loss associated with controlled trajectory $\mathbf{x}(t)$ relative to reference trajectory $\mathbf{x}_{\mathrm{R}}(t)$.
	Choice of parameterization.
	Choice of training set. Explain choice in one paragraph.

We will check that your answers are correct. If they are not, we will get back to you and ask you to correct them. As long as you submit responses, you get an A for the assignment. It counts for 10% of your lab grade.