CHEM-E7225 Advanced Process Control D — Winter 2026

Lecture 10 – Model Predictive Control: Output feedback

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

In this final lecture, we present an output-feedback control design consisting of the interconnection of a model-predictive controller and a moving-horizon estimator. We also present some future directions for improving our control designs.

1 Output-feedback receding-horizon control

The model-predictive controllers we studied are state-feedback controllers, that is, they operate by directly measuring the state of the system to be controlled. This requirement is quite strict. In many applications, due to technological and/or economical reasons, we only have enough instrumentation to partially observe the state. Moreover, our models are imperfect and subject to disturbances/noise that prevents us from predicting the state exactly.

In this course, we tackled these issues by designing *state estimators*, devices responsible for determining the state in real time based on measurement data. The natural extension of a state-feedback controller to output-feedback is thus having the controller use the state signal produced by the estimator (Figure 1).

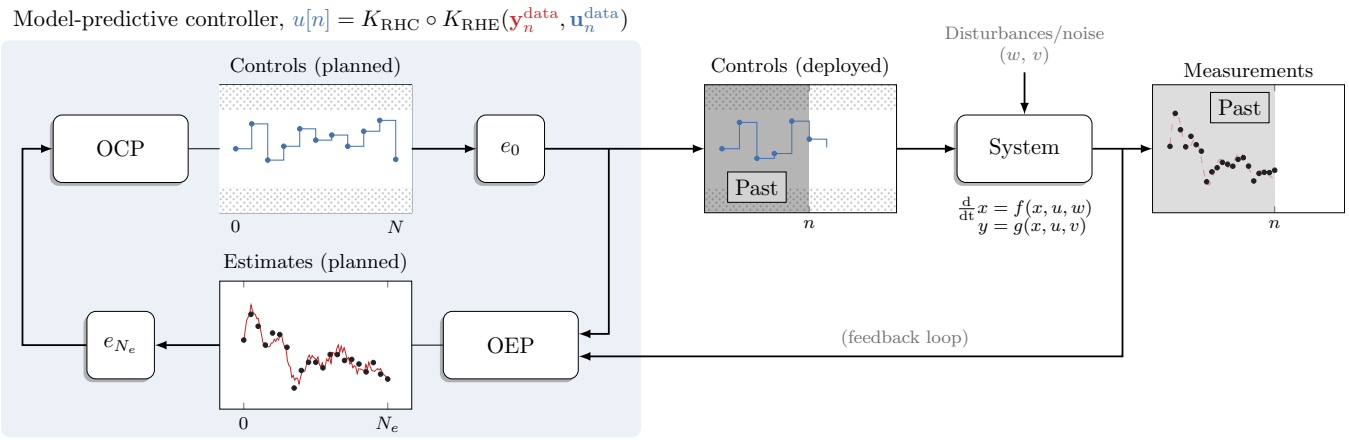

Figure 1. Diagram of an output-feedback controller consisting of a state-estimator coupled to a state-feedback controller.

Formally, in a digital setup, we have the output-feedback control policy

$$u[n] = K_c \circ K_e\big(\mathbf{y}_n^{\text{data}},\ \mathbf{u}_n^{\text{data}}\big) \tag{1}$$

given the control and estimation policies $K_c(\cdot)$ and $K_e(\cdot)$, respectively, and the measurement $\mathbf{y}_n^{\text{data}}$ and control $\mathbf{u}_n^{\text{data}}$ data collected until the nth timestep. Importantly, the interconnection leads to a recursive controller: to compute the actions $u[n]$, the output-feedback controller (specifically, its underlying estimator) must also know the past actions $\mathbf{u}_n^{\text{data}}$ applied to the system. Summarizing, in an output-feedback policy, we (i) collect the past data $(\mathbf{y}_n^{\text{data}}, \mathbf{u}_n^{\text{data}})$; (ii) estimate the current state $\hat{x}[n] = K_e(\mathbf{y}_n^{\text{data}}, \mathbf{u}_n^{\text{data}})$; then (iii) compute the control action $u[n] = K_c(\hat{x}[n])$. In the case of receding-horizon setpoint-tracking policies, we have that

$$u[n] = u_{\text{ref}} + K_{\text{RHC}} \circ K_{\text{RHE}}\big(\mathbf{y}_n^{\text{data}} - y_{\text{ref}},\ \mathbf{u}_n^{\text{data}} - u_{\text{ref}}\big). \tag{2}$$

A noteworthy feature here is that the steady-state, x_{ref}, is "masked" by the interconnection of the controller with the estimator. Specifically, this is the result of plugging $\hat{x}[n] = x_{\text{ref}} + K_{\text{RHE}}(\cdot)$ into $u[n] = u_{\text{ref}} + K_{\text{RHC}}(\hat{x}[n] - x_{\text{ref}})$. An interesting consequence is that a setpoint change can be enforced by simply changing y_{ref}, without the need for redefining the control/estimation policies–although the tracking might have an offset if the change forces the system to escape the region of attraction of the equilibrium point $(x_{\text{ref}}, u_{\text{ref}})$.

In the following, we overview the MPC policy $K_{\text{RHC}}(\cdot)$ and MHE policy $K_{\text{MHE}}(\cdot)$ individually, and present any changes in their formulation caused by this interconnected architecture.

1.1 Model-predictive controller

Let's consider a L-MPC policy $K_{\text{RHC}}(\hat{x}[n] - x_{\text{ref}}) = e_0 \circ \texttt{OCP}(\hat{x}[n] - x_{\text{ref}})$ for operating a nonlinear system subjected to disturbances, with $\texttt{OCP}(\cdot)$ being a solution to the linear-quadratic optimal control problem

$$\underset{x^\delta[\cdot],\ u^\delta[\cdot]}{\text{minimize}} \quad \sum_{k=0}^{N-1} \begin{bmatrix} x^\delta[k] \\ u^\delta[k] \end{bmatrix}^\mathsf{T} \begin{bmatrix} Q & S \\ S^\mathsf{T} & R \end{bmatrix} \begin{bmatrix} x^\delta[k] \\ u^\delta[k] \end{bmatrix} + x^\delta[N]^\mathsf{T} Q_f x^\delta[N] \tag{3a}$$

$$\text{subject to} \quad x^\delta[k{+}1] = A_{\Delta t} x^\delta[k] + B_{\Delta t} u^\delta[k] + w[k], \quad k = 0, \dots, N-1, \tag{3b}$$

$$x^\delta_{\min} \preceq x^\delta[n] \preceq x^\delta_{\max}, \qquad k = 0, \dots, N-1, \tag{3c}$$

$$u^\delta_{\min} \preceq u^\delta[n] \preceq u^\delta_{\max}, \qquad k = 0, \dots, N-1, \tag{3d}$$

$$x^\delta[0] = \hat{x}^\delta[n]. \tag{3e}$$

The problem is expressed in deviation variables with $(A_{\Delta t}, B_{\Delta t})$ being the linearization (with subsequent discretization) of the dynamics $f(\cdot)$ around an operating point $(x_{\text{ref}}, u_{\text{ref}})$. In this context, the L-MPC is problem is structurally similar to that from previous lectures. However, there are two differences. Firstly, the boundary condition (Eq. 3e) is now expressed in terms of the state estimate $\hat{x}^\delta[n]$. Secondly, the dynamic constraints (Eq. 3b) now include the random disturbance process $w[\cdot]$. This is an issue, because it leads to Problem (3) becoming a stochastic optimization problem which require special solution methods.

An approach to deal with the disturbances is based on the *certainty-equivalence principle*: we assume $(A_{\Delta t}, B_{\Delta t})$ to represent the *nominal* dynamics of the system, and substitute the disturbance process with its constant mean value, $w[k] = \mu_w$. In the (common) case of zero-mean Gaussian disturbances $w[n] \sim \mathcal{N}(0, S_w^2)$, Problem (3) is exactly the LQ Regulator problem of previous lectures. Thus, under this approach, the state-feedback receding-horizon control policy remains unchanged; we simply need to connect it to a state estimator.

1.2 Moving-horizon estimator

Let's consider a L-MHE policy $K_{\text{RHE}}(\cdot) = e_{N_e} \circ \texttt{OEP}(\cdot)$ for estimating the state of a partially-observed nonlinear system, with $\texttt{OEP}(\cdot)$ being a solution to the (unconstrained) linear-quadratic optimal estimation problem

$$\underset{x^\delta[\cdot],\ w^\delta[\cdot]}{\text{minimize}} \quad \sum_{k=0}^{N_e-1} \begin{bmatrix} \epsilon(x^\delta[k]) \\ w^\delta[k] \end{bmatrix}^\mathsf{T} \begin{bmatrix} Q_e & 0 \\ 0 & R_e \end{bmatrix} \begin{bmatrix} \epsilon(x^\delta[k]) \\ w^\delta[k] \end{bmatrix} + \epsilon(x^\delta[N_e])^\mathsf{T} Q_e \epsilon(x^\delta[N_e]) \tag{4a}$$

$$\text{subject to} \quad x^\delta[k{+}1] = A_{\Delta t} x^\delta[k] + B_{\Delta t} u_n^{\text{data},\delta}[k] + w[k], \quad k = 0, \dots, N-1, \tag{4b}$$

given the auxiliary measurement error signal $\epsilon(x^\delta[k]) = C x^\delta[k] - y_n^{\text{data},\delta}[k]$. The problem is expressed in deviation variables with $(A_{\Delta t}, B_{\Delta t})$ being the same linearization (with subsequent discretization) of the dynamics $f(\cdot)$ around the operating point $(x_{\text{ref}}, u_{\text{ref}})$ used for the L-MPC. We remark that the L-MPC and L-MHE both using the same linear(ized) model is not a requirement, but a convenient design choice.

In the output-feedback context, the L-MHE is very similar to its standalone formulation. The only difference is that now it depends on past measurement data (for the objective) and also on *past input data (for the constraints)*,

$$\mathbf{y}_n^{\text{data}} = (y[n{-}N_e],\ y[n{-}N_e{+}1],\ \dots, y[n{-}1], y[n]); \tag{5}$$

$$\mathbf{u}_n^{\text{data}} = (u[n{-}N_e],\ u[n{-}N_e{+}1],\ \dots, u[n{-}1]), \tag{6}$$

The measurement and control action datasets can be updated recursively as $\mathbf{y}_n^{data} = [\mathbf{y}_{n-1}^{data}[2{\rightsquigarrow}N_e] \ \ y[n]]$ *and* $\mathbf{u}_n^{data} = [\mathbf{u}_{n-1}^{data}[2{\rightsquigarrow}N_e{-}1] \ \ u[n{-}1]]$, *where* N_e *is the number of datapoints in the dataset.*

or, in more compact form, $u_n^{\text{data}}[k] = u[n{-}N_e{+}k]$ and $y_n^{\text{data}}[k] = y[n{-}N_e{+}k]$. The necessity for including the term $B_{\Delta t} u_n^{\text{data}}[k]$ $(k = 0, \dots, N_e{-}1)$ in the dynamic constraints (Eq. 4b) is self-evident: since the estimator is reconstructing states that happened in the past, it should also account for the input actions applied to the system at those timesteps.

2 †Extra: Future directions

If you reached this point, congratulations! Together, we have built fundamental skills on designing and implementing optimal output-feedback controllers for nonlinear multivariable dynamic systems. Although we have come a long way, there are still many exciting directions to improve our designs and deploy more advanced controllers. In the following, we explore two of these directions which you can follow using the skills we learned throughout the course. We refer to the supporting bibliography for more details.

Optimal steady-state computation / real-time optimization (RTO)

The output-feedback MPC we designed assumes the availability of an operating point $(x_{\text{ref}}, u_{\text{ref}})$ which is an equilibrium (i.e., $0 = f(x_{\text{ref}}, u_{\text{ref}})$). In reality, it is more common to have an *output setpoint* y_{ref} which we want to enforce. Therefore, it is possible to extend our control architecture to include a module responsible for *steady-state computation*, also known as *real-time optimization* (RTO) in some books. The extension is depicted in Figure 2.

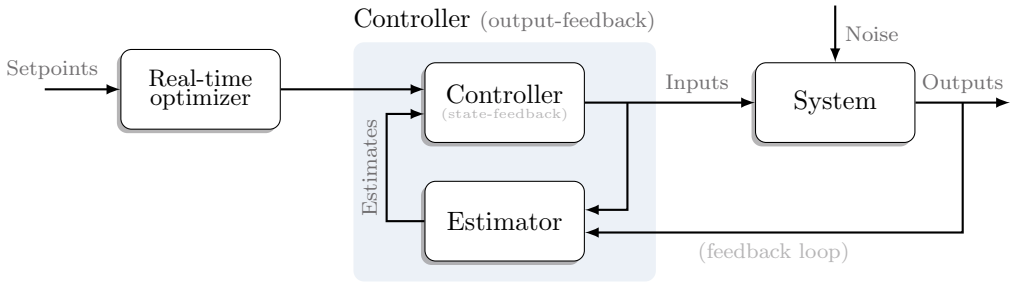

Figure 2. Diagram of an output-feedback controller equipped with a steady-state computation module.

Formally, the steady-state computation module implements the relation

$$(x_{\text{ref}}, u_{\text{ref}}) = K_{\text{SS}}(y_{\text{ref}}) \tag{7}$$

given some policy function K_{SS}. Using our already well-established optimization frameworks, a possible design choice is to have $K_{\text{SS}}(y_{\text{ref}})$ be a solution to the static optimization problem

$$\begin{aligned}
\underset{x_{\text{ref}},\ u_{\text{ref}}}{\text{minimize}} \quad & (g(x_{\text{ref}}) - y_{\text{ref}})^{\mathsf{T}} Q_{\text{ref}} (g(x_{\text{ref}}) - y_{\text{ref}}) && \text{(8a)} \\
\text{subject to} \quad & 0 = f(x_{\text{ref}}, u_{\text{ref}}), && \text{(8b)} \\
& H_x x_{\text{ref}} + H_u u_{\text{ref}} \preceq h, && \text{(8c)}
\end{aligned}$$

where we directly use the nonlinear model functions $f(\cdot)$ and $g(\cdot)$. A solution to this problem is an equilibrium pair $(x_{\text{ref}}, u_{\text{ref}})$ such that $g(x_{\text{ref}}) \approx y_{\text{ref}}$ as close as possible. The tuning matrix Q_{ref} can be used to prioritize matching specific output components. Finally, the inequality constraints ensure that the obtained steady-states are feasible.

The above module is a good addition to a setpoint-tracking MPC, as it simplifies it operation. However, we remark that the problem we formulate above is *nonconvex* and thus suffers from performance and robustness issues. Regardless, since the problem is static, it is not as high-dimensional as OCPs/OEPs, and thus it is still computationally practical.

Nonlinear model-predictive controllers

An extension which is readily applicable to our framework is having the MPC based on the optimal control problems

$$\begin{aligned}
\underset{x[\cdot],\ u[\cdot]}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \begin{bmatrix} x[k] \\ u[k] \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} Q & S \\ S^{\mathsf{T}} & R \end{bmatrix} \begin{bmatrix} x[k] \\ u[k] \end{bmatrix} + x[N]^{\mathsf{T}} Q_f x[N] && \text{(9a)} \\
\text{subject to} \quad & x[k{+}1] = f_{\Delta t}(x[k], u[k], w[k]), \quad k = 0, \dots, N-1, && \text{(9b)} \\
& h(x[n], u[n]) \preceq 0, \quad k = 0, \dots, N-1, && \text{(9c)} \\
& x[0] = \hat{x}[n]. && \text{(9d)}
\end{aligned}$$

Problem (9) considers the nonlinear transition function $f_{\Delta t}(\cdot)$ (e.g., a RK4 integrator) instead of linearized dynamics, and the path constraints (Eq. 9c) are given in general nonlinear form. In this formulation, the problem is nonconvex. Thus, it might be computationally expensive or prone to return locally optimal solutions. On the other hand, since the nonlinear model is a more accurate representation of the system, this is a more powerful controller in the sense that it is more flexible than linear controllers. For instance, nonlinear MPCs can perform initialization/recovery routines, which are much more complicated (or impossible) to implement using linearized dynamics. Therefore, on control design, there is always a tradeoff between performance/reliability/simplicity vs. accuracy/flexibility/complexity when discussing which dynamic model to consider. In reality, the best control architecture is whichever of the two works.

References

[1] J. T. Betts. *Practical methods for optimal control and estimation using nonlinear programming*. Advances in Design and Control. SIAM, 2010. ISBN: 978-0-89871-688-7.

[2] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. ISBN: 978-1-13-906175-9.

[3] 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.