CHEM-E7225 Advanced Process Control D — Winter 2026

# Lecture 01 – Introduction to Advanced Process Control

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

## Welcome to CHEM-E7225 Advanced Process Control!

In this course, we study numerical optimization and its application to the control of process systems. The focus is on multivariable (possibly large-scale) chemical systems whose dynamical behavior is described by a mathematical model. In this direction, we use techniques from *dynamical systems theory* and *numerical optimization* to design a *controller*, a device capable of automatically operating the process system to achieve a prescribed objective.

## 1 The "Big Picture" – The principle of feedback and optimal control

Before diving into the technicalities, let's first understand the "big picture" behind feedback control.

The starting point is the following: we have a process designed to manufacture some product–for illustrative purposes, consider a reactor converting some input feedstock (which we can manipulate) into an output product (which we can monitor). The system is dynamical, that is, *its output evolves in time in response to the inputs*. In principle, if everything works as idealized, we should have a clear idea of how to operate the process (i.e., manipulate its inputs) to reach a *steady-state* in which it produces the desired output. Unfortunately, this idealized scenario is not realistic. Models are imperfect and there is always some external *noise/disturbances* affecting the system, eventually driving it away from the desired state and/or causing constraint violations (Figure 1). For our example, these can be impurities in the feed and temperature/pressure perturbations inside the reactor. The process thus will not perform as expected unless we *control its outputs* to account for model uncertainty and to reject the effects of the noise/disturbances.

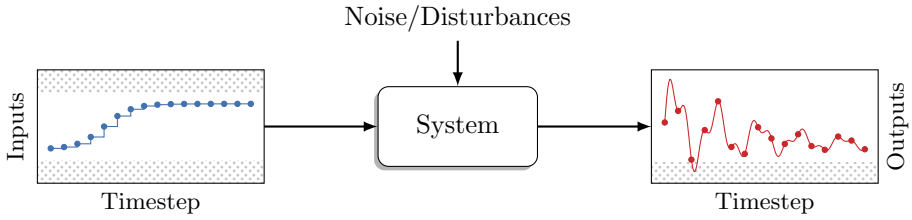



**Figure 1.** In *open-loop*, we design an input signal expecting the process system to respond according to our model. In reality, the system is affected by external noise/disturbances not accounted by the model, making its true evolution deviate from the setpoint.

The noise/disturbances cannot be manipulated, and, because they are understood to be stochastic, they also cannot be anticipated. Therefore, we need a mechanism to react to changes in the output and automatically adjust the inputs accordingly. In this course, we achieve this self-regulating behavior using a *feedback controller*: a device that receives a reference signal and the output signal, then decides which input signal must be deployed for the process to achieve its operational objectives. This feedback interconnection (a.k.a. "closed-loop control") is depicted in Figure 2.

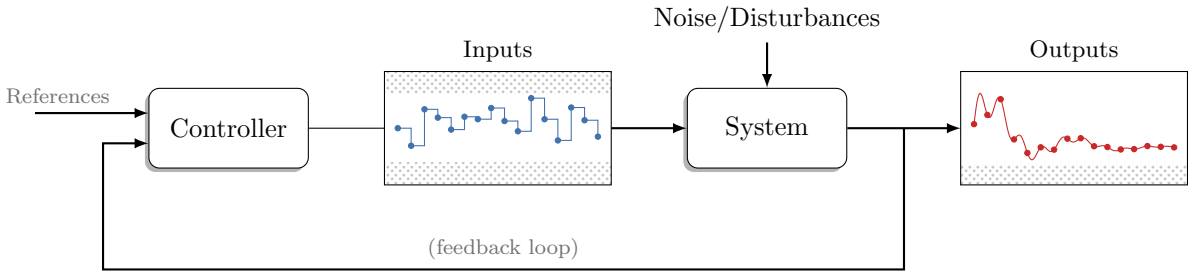



**Figure 2.** Feedback interconnection: The inputs are computed by a digital device, the controller, to correct deviations from the references caused by the noise/disturbances affecting the system. Note that closed-loop performance is better, but still affected by the noise.

Let's consider some examples of process systems in which feedback control plays an important role.

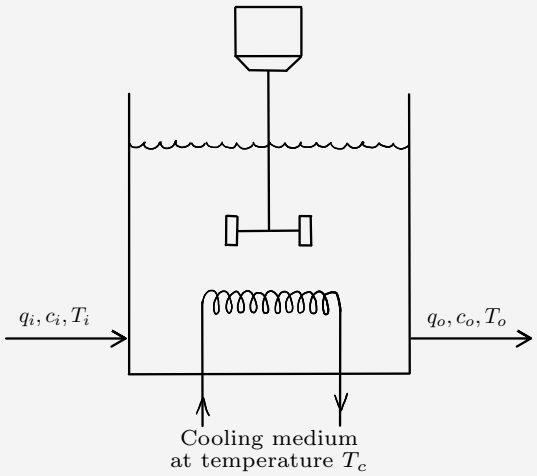
**Example 1.1** ***A continuously-stirred tank reactor [5]***

A continuously-stirred tank reactor (CSTR) is a system which implements a (bio)chemical reaction network in which a feedstock is converted into some desired product.

This system generalizes many continuous (bio)chemical process in industry. The control problem is often to ensure an output of products with consistent quality. The controller might be also subject to technical/economical limitations (e.g., valves capacities, or "unsafe" operating regions). In most applications, the relevant process variables are:

**Inputs:** The influent/effluent flow-rates ($q_i$ and $q_o$) and cooling medium temperature ($T_c$)

**Outputs:** The liquid volume ($V$), its temperature ($T_o$), and its concentrations of products ($c_o$)




The influent composition ($c_i$) and temperature ($T_i$) are non-manipulable: they are often treated as disturbances. The dynamics of this class of reactors are modelled using mass-action kinetics and energy conservations laws.

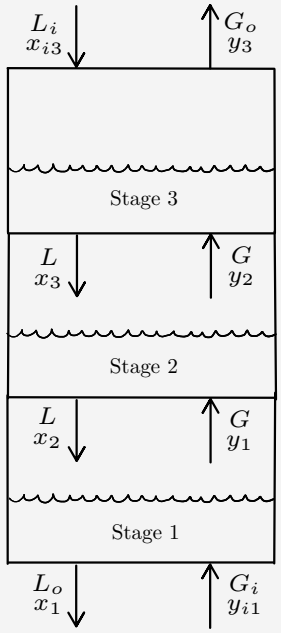
**Example 1.2** ***Staged processes (distillation, absorption, extraction, ...) [5]***

A common class of process systems, particularly in separation processes, are multistage (or staged) systems. In these processes, materials are brought into contact to reach a desired equilibrium between individual phases. Staged systems include distillation, absorption, and extraction units. The feed/outflow of the system can be at any of the stages, which are connected either horizontally or vertically. In general, the control problem is to regulate the phase equilibria while simultaneously improving the separation efficiency.

In liquid-gas $N$-staged systems, the relevant process variables are

**Inputs:** Inflow/outflow of liquids and gas ($L_{in}/L_{on}$ and $G_{in}/G_{on}$, respectively) at some stages $n \in 0, \dots, N$

**Outputs:** Liquids and gas concentrations ($x_n$ and $y_n$, respectively) at some stages $n \in 0, \dots, N$




The feed liquid-gas compositions ($x_i$ and $y_i$, respectively) are often non-manipulable and treated as disturbances. The dynamics of this class of processes are modelled using material and energy balance equations for each stage, individually.

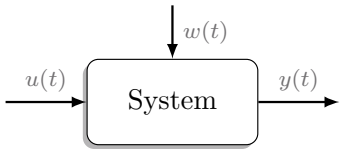
Note that we usually cannot manipulate and/or measure all the stages in the system, but only a subset of them. Thus, such systems are said to be underactuated/underobserved. Multistage systems are multivariable systems of notorious complexity and often require advanced process control to achieve a satisfactory performance.

A question arises: How to design the feedback controller? In our case, we are interested in design methods that can address multi-input multi-output systems with operational constraints (e.g., safety requirements or actuator limits), and that optimize some performance criteria. Arguably, we should exploit our knowledge about the system (that is, its dynamical model) to formulate a control policy that is guaranteed to satisfy these specifications. Therefore, in this course, we take this model-based approach and study state-of-the-art methods to design feedback controllers capable of (i) *optimally achieving a desired operational objective*, while (ii) *explicitly enforcing any operational restrictions.*

# 2 Course Roadmap

In the following, we outline the main topics we study in this course. We only present the basic ideas and concepts, since they will be explained in more details in the next lectures.

## 2.1 Dynamical models




The central element in model-based control is a mathematical description of the dynamics of the system we are operating. In this course, we focus on state-space representations of the form

$$\tfrac{\mathrm{d}}{\mathrm{dt}}x(t) = f(x(t), u(t), w(t)); \tag{1a}$$
$$y(t) = g(x(t), u(t), w(t)), \tag{1b}$$

where $x(t) \in \mathbb{R}^{N_x}$ is the process state, $y(t) \in \mathbb{R}^{N_y}$ is the output (or measurement), $u(t) \in \mathbb{R}^{N_u}$ are the control inputs deployed to the system, and $w(t) \in \mathbb{R}^{N_w}$ are the noise/disturbances, at time $t \in \mathbb{R}$. Equation (1a) is an ordinary differential equation describing how the state *changes in time* (that's the meaning of the differential operator $\frac{\mathrm{d}}{\mathrm{dt}}$). Equation (1b) is an algebraic equation describing how the state (plus inputs and noise) is emitted as the outputs. The state function $f$ often arises from mass/energy/momentum balance equations of the form

$$\underbrace{[\text{Stuff accumulated}]}_{\frac{\mathrm{d}}{\mathrm{dt}}x(t)} = \underbrace{[\text{Stuff in}] - [\text{Stuff out}] \pm [\text{Stuff generated/consumed}]}_{f(x(t),u(t),w(t))}.$$

The output function $g$ comes directly from the sensor layout (i.e., what can be measured, and how).

The state-space model allows us to investigate dynamical properties of the physical system (stability, detectability, etc) and, most importantly, to predict its state/output response to the input signal. Specifically, we can simulate the state response $x(t)$ over time $t \in [t_0, t_f]$ given an initial state $x(t_0)$ and inputs $u(t)$ and $w(t)$ over $t \in [t_0, t_f]$. Consequently, the outputs $y(t)$ over $t \in [t_0, t_f]$ can also be predicted. The controllers we design exploit this predictive capability to determine the best control actions, given a desired operational objective and a set of operational constraints.

A fundamental class of state-space models are linear time-invariant (LTI) models,

$$\tfrac{\mathrm{d}}{\mathrm{dt}}x(t) = Ax(t) + B_u u(t) + B_w w(t); \tag{2a}$$
$$y(t) = Cx(t) + D_u u(t) + D_w w(t), \tag{2b}$$

in which the dynamics $f(\cdot)$ are defined by the matrices $(A, B_u, B_w)$ and the measurement process $g(\cdot)$ is defined by the matrices $(C, D_u, D_w)$. The properties and numerics of LTI systems are well-understood, making them an essential framework under which to design model-based controllers. While they are clearly less accurate than general, nonlinear models, we will see that even these simple linear models are a powerful tool for controlling complex dynamical systems.

In the course, we will overview the fundamentals of state-space models and their properties. The student will learn how to understand state-space models, analyze its stability properties, obtain linear approximations (when needed), and simulate their time evolution using numerical methods.

## 2.2 Optimal feedback control


In this course, we consider digital controllers that compute the inputs with a *control frequency* of $1/\Delta t$. A digital feedback controller implements a *control policy*,

$$u[n] = K(y[n], r[n]) \tag{3}$$

describing how the inputs are computed based on the current value of the outputs and references. The control policy $K(\cdot)$ is our design, and a specific structure instantiates a specific *class of controllers*. We use square brackets $[\cdot]$ to indicate that the signals are now sampled in discrete-time (that is, they map integers to vectors), as illustrated in Figure 3. In special, a continuous-time signal $u(\cdot)$ can be reconstructed from the discrete-time signal $u[\cdot]$ using, for instance, a zero-order hold filter such that

$$u(t) = u[n] \quad \text{for} \quad t \in [n\Delta t,\ (n{+}1)\Delta t). \quad \text{or, in compact form, } u(t) = u\big[\texttt{floor}(t/\Delta t)\big].$$

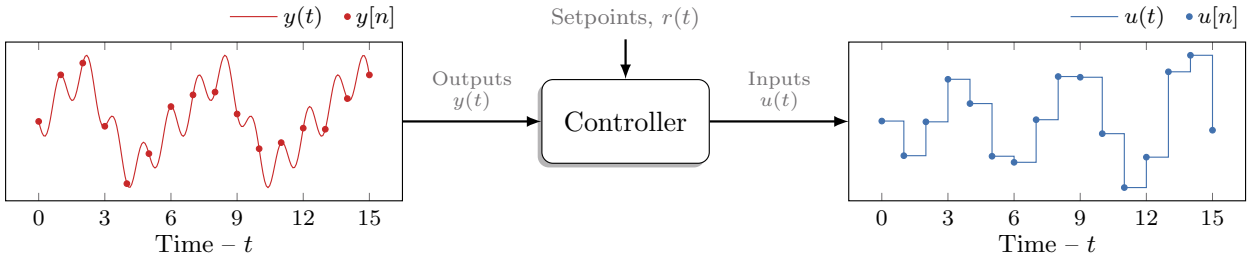



**Figure 3.** A digital controller operates by measuring the outputs and computing the inputs with a frequency of $1/\Delta t$. The continuous-time signal $u(t)$ is reconstructed by a zero-order hold filter, leading to a piecewise-continuous signal. In the above example, $\Delta t = 1$ [units-of-time].

A common rule of thumb is that controller performance is proportional to the control frequency $1/\Delta t$, which is limited by the computational capacity of the physical equipment (e.g., some microcontrollers max out at 1 kilohertz). While our systems are modelled in continuous-time, our control methods will always be designed in discrete-time. The interconnection of a controller (a digital device) with a physical system is sometimes known as a *cyber-physical system*.

**Remark.** *A controller also admits a state-space representation, in which an internal state $z_c[\cdot]$ acts as a "memory" or "buffer" (see the short paper [6]). However, the policy formulation (Eq. 3) is more common in literature/industry.*

### 2.2.1 Receding-horizon control

In the following, consider that the state is fully measurable, that is, $y(t) = x(t)$. We are interested in feedback controllers which are both optimal and capable of dealing with multivariable constrained systems. The *model-predictive controller* (MPC) is a state-of-the-art class of optimal controllers, based on the receding-horizon control policy

$$u[n] = e_0 \circ \texttt{OCP}(x[n]) \tag{4}$$

in which $e_0(\cdot)$ is a selector operator and `OCP` is the optimizer mapping

$$\texttt{OCP}(x_0) = \arg\min_{\mathbf{u}} \big\{ V_N(x_0, \mathbf{u}) \mid \mathbf{u} \in \mathcal{U}_N(x_0) \big\} \tag{5}$$

defined for a sequence of actions $\mathbf{u} = (u[0], \dots, u[N-1])$, and $N$-horizon value $V_N(\cdot)$ and constraint $\mathcal{U}_N(\cdot)$ functions

$$V_N(x_0, \mathbf{u}) = \sum_{k=0}^{N-1} L\big(x[k], u[k]\big) + V_f\big(x[N]\big) \tag{6a}$$

$$\mathcal{U}_N(x_0) = \big\{ \mathbf{u} \in \mathbb{R}^{NN_u} \mid h(x[k], u[k]) \leq 0, \quad h_f(x[N]) \leq 0, \quad \forall k \in [0, N-1] \big\}. \tag{6b}$$

The value functional is the performance metric that the controller must minimize, in a stagewise fashion (e.g., distance to a setpoint, energy expenditure, economic costs, and so on). The path constraints are restrictions that the controller must satisfy/enforce while under operation (e.g., actuator limits, safety requirements, cyclic constraints, and so on).

The MPC policy (Eq. 4) thus works as follows: at each $n$th control cycle, the controller

(i) computes an optimal sequence of actions $\mathbf{u}^\star = (u^\star[0], \dots, u^\star[N-1])$ that minimizes the value function $V(\cdot)$ given the current state $x_0 = x[n]$, then

(ii) extracts and deploys only the first optimal action to the process, that is, $u[n] = e_0(\mathbf{u}^\star) = u^\star[0]$.

The remaining controls $(u^\star[1], \dots, u^\star[N-1])$ are discarded–the idea is that we must recomputed them at each cycle to account for the (likely) changes in the state caused by the disturbances. In practice, at each control cycle, the controller solves the *nonlinear optimization problem* (a.k.a. nonlinear program),

$$\underset{x[\cdot],\ u[\cdot]}{\text{minimize}} \quad \sum_{k=0}^{N-1} L\big(x[k], u[k]\big) + V_f\big(x[N]\big) \tag{7a}$$

$$\underset{k=0,N-1}{\text{subject to}} \quad x[k+1] = f_{\Delta t}\big(x[k], u[k], w[k]\big), \quad x[0] = x[n], \tag{7b}$$

$$h\big(x[k], u[k]\big) \leq 0, \quad h_f\big(x[N]\big) \leq 0, \tag{7c}$$

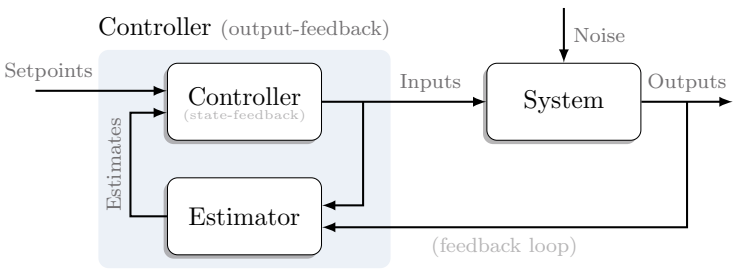
where $f_{\Delta t}(\cdot)$ is a *transition function* describing how the states $x[k]$ $(k = 1, \dots, N-1)$ are recursively computed.

The art of designing an MPC is centred on choosing the appropriate control horizon $N$, frequency $1/\Delta t$, cost functions $\{L(\cdot), V_f(\cdot)\}$, and constraints functions $\{h(\cdot), h_f(\cdot)\}$, such that the closed-loop process system behave as desired. A control engineer is also responsible for implementing a *numerical method* for solving Problem (7), as analytical solutions are rarely possible. Moreover, note that the value and constraint functions depend explicitly only on the current state $x_0 = x[n]$, since future states can be computed recursively starting from $x_0$. Because solving Problem (7) depends on fixing a parameter, the initial state $x[0] = x[n]$, it is said to be a *parametric* optimization problem.

**Remark.** *Actually, the mapping* $\texttt{OCP}(\cdot)$ *also depend explicitly on the reference signal* $r[\cdot]$*. We omit this dependency to simply notation, since, as we will see later, setpoint tracking can be encoded directly on the predictive model.*

### 2.2.2 Receding-horizon estimation

The previous assumption of full-state measurability is very strong: in practice, because of technological/economical limitations, the state can only by partially measured. In such cases, we are restricted to output-feedback policies. However, the MPC formulated in Eq. (4) is implementing a state-feedback policy. A question arises: How to enable output-feedback model-predictive control? In this course, we design output-feedback MPCs by equipping them with state estimators, devices that reconstruct the state in real-time from measurement data. We consider *moving-horizon estimators* (MHE), which are defined by receding-horizon estimation policies of the form




$$\hat{x}[n] = e_{N_e} \circ \texttt{OEP}(\mathbf{y}_{\text{data}}) \tag{8}$$

in which $e_{N_e}(\cdot)$ is a selector operator and $\texttt{OEP}(\cdot)$ is the optimizer mapping

$$\texttt{OEP}(\mathbf{y}_{\text{data}}) = \arg\min_{\mathbf{x}} \left\{ V^e_{N_e}(\mathbf{y}_{\text{data}}, \mathbf{x}) \mid \mathbf{x} \in \mathcal{X}_{N_e} \right\} \tag{9}$$

defined for measurement history $\mathbf{y}_{\text{data}} = (y[n{-}N_e], \dots, y[n])$, sequence of state estimates $\mathbf{x} = (x[0], \dots, x[N_e])$, and the $N_e$-horizon value function $V^e_{N_e}(\cdot)$ and the constraint set $\mathcal{X}_{N_e}$

$$V^e_{N_e}(\mathbf{y}_{\text{data}}, \mathbf{x}) = V^e_0\big(x[0]\big) + \sum_{k=1}^{N_e} L^e\big(y_{\text{data}}[k], x[k]\big), \tag{10a}$$

$$\mathcal{X}_{N_e} = \left\{ \mathbf{x} \in \mathbb{R}^{(N_e+1)N_x} \mid h^e_0(x[0]) \leq 0, \quad h^e(y_{\text{data}}[k], x[k]) \leq 0, \quad \forall k \in [1, N_e] \right\}. \tag{10b}$$

Here, the value function is an information metric that the estimator must optimize (e.g., negative likelihood), while the path constraints encode the feasible set of the estimates (e.g., known state/disturbance bounds).

The MHE policy (Eq. 8) thus works as follows: at each $n$th estimation cycle, the estimator

(i) computes an optimal sequence of estimates $\mathbf{x}^\star = (x^\star[0], \dots, x^\star[N_e])$ that minimizes the value function $V^e_{N_e}(\cdot)$ given the past history of measurement data $\mathbf{y}_{\text{data}} = (y[n{-}N_e], \dots, y[n])$, then

(ii) extracts and deploys only the last optimal estimate to the MPC, that is, $\hat{x}[n] = e_{N_c}(\mathbf{x}^\star) = x^\star[N_c]$.

The remaining estimates $(x^\star[0], \dots, x^\star[N_e{-}1])$ are discarded–the idea is that they must be recomputed at each cycle, to account for the new measurements. Similar to before, at each cycle the estimator solves the nonlinear program

$$\underset{x[\cdot],\ w[\cdot]}{\text{minimize}} \quad V^e_0\big(x[0]\big) + \sum_{k=1}^{N_e} L^e\big(y_{\text{data}}[k], x[k]\big) \tag{11a}$$

$$\underset{k=0,\dots,N_e-1}{\text{subject to}} \quad x[k{+}1] = f_{\Delta t}\big(x[k], u[k], w[k]\big), \tag{11b}$$

$$h^e_0\big(x[0]\big) \leq 0, \quad h^e\big(x[k], u[k]\big) \leq 0, \tag{11c}$$

The cost $\{V^e_0(\cdot), L^e(\cdot)\}$ and constraint $\{h^e_0(\cdot), h^e(\cdot)\}$ functions often arise naturally from some statistical assumption on the noise/disturbances $w[\cdot]$, which is the source of uncertainty in our dynamical model. Since the state estimator reconstructs a signal from noise-corrupted measurements, it is also referred to as a *filter*. Finally, we note that the

control and estimation cycles do not need to coincide, that is, we could have the MHE producing estimates at a different frequency $1/\Delta t_e > 0$. However, there is not much practical reason to consider such a setup (aside from when $\Delta t_e < \Delta t$ due to technological limitations). Thus, unless stated otherwise, we will always assume $\Delta t_e = \Delta t$.

The MHE is structurally identical to the MPC; in fact, one is said to be a "dual" to the other. The state estimation problem, however, is concerned with a *past* receding horizon, and the goal is to find a trajectory $\mathbf{x}$ that best explains the data $\mathbf{y}_{\text{data}}$, while still feasible according to the constraints $\mathcal{X}_{N_e}$. Moreover, the input data to the MPC policy is only a single vector, the current state $x[n]$, while the input data to the MHE is a sequence of vectors, the measurements history $\mathbf{y}_{\text{data}}$. Because of this, the MPC is said to be a *memoryless* device, while the MHE is a *memorybased* device.

**Remark.** *The value function $V^e_{N_e}(\cdot)$ also depend on the past values of the input signal $u[\cdot]$. We omit this dependency both to simply notation and because, as we'll see, the predictive model can be rewritten to encode the inputs.*

**Remark.** *In practice, the MPC and MHE can be designed independently. For this reason, the course will be focused on state-feedback problems with $y(t) = x(t)$, and we will only discuss output feedback in the latter lectures.*

# 3 Course Program and Assessment

The course is comprised of 12 lectures ($\approx$ 24h) and 6 exercise sessions ($\approx$ 12h), together with home assignments and independent study ($\approx$ 80h). We will follow the (tentative) program outlined below.

| | | |
|---|---|---|
| **Week 1 – Introduction + Dynamical Systems** | | |
| L01 (07/Jan, Ke5) | Introduction to Advanced Process Control | |
| L02 (09/Jan, Ke5) | Dynamic Systems: Models and simulation | |
| E01 (09/Jan, B016) | Simulation of Dynamical Process Systems | (HW Deadline: 18/Jan) |
| **Week 2 – Nonlinear Programming Basics** | | |
| L03 (14/Jan, Ke5) | Nonlinear Programming (A) | |
| L04 (16/Jan, Ke5) | Nonlinear Programming (B) | |
| E02 (16/Jan, B016) | Nonlinear Programming for Process Control | (HW Deadline: 25/Jan) |
| **Week 3 – Model Predictive Control, unconstrained problems** | | |
| L05 (21/Jan, Ke5) | Model Predictive Control (A) | |
| L06 (23/Jan, Ke5) | Model Predictive Control (B) | |
| E03 (23/Jan, B016) | Optimal Control of Process Systems | (HW Deadline: 01/Feb) |
| **Week 4 – Model Predictive Control, constrained problems** | | |
| L07 (28/Jan, Ke5) | Model Predictive Control - Barrier methods (A) | |
| L08 (30/Jan, Ke5) | Model Predictive Control - Barrier methods (B) | |
| E04 (30/Jan, B016) | Optimal Control of (Constrained) Process Systems | (HW Deadline: 08/Feb) |
| **Week 5 – Moving Horizon Estimation + Output-feedback MPC** | | |
| L09 (04/Feb, Ke5) | Moving Horizon Estimation | |
| L10 (06/Feb, Ke5) | Nonconvex MPC + MHE | |
| E05 (06/Feb, B016) | Optimal Estimation of Process Systems | (HW Deadline: 15/Feb) |
| **Week 6 – Free Topic (Tips&Tricks, Seminars, Advanced Algorithms, ...)** | | |
| L11 (11/Feb, Ke5) | TBA | |
| L12 (13/Feb, Ke5) | TBA | |
| E06 (13/Feb, B016) | Final Assignment | (FA Deadline: 01/Mar) |

The primary study material are these lecture notes. Additional study materials are listed in the bibliography section below. If you have any questions during your studies, feel free to reach out via email (`otacilio.bneto@pm.me`) or during office hours (Room E301, Kemistintie 1, Mon-Fri 16:00 – 18:00).

The exercises/homeworks are mainly computer exercises. The material is based on MATLAB/CasADi. The students are free (and encouraged) to use any other programming language that they prefer (e.g., Python), but **must inform the course staff if so**. Each exercise section will be divided in two parts: (i) a "hands on" part ($\approx$ 1h), in which we solve some warm-up exercises together, and (ii) a "free time" part ($\approx$ 45mins), in which the students are free to work in any open assignments with the help of the course staff.

The course grade is based on 5 weekly homeworks (40%) and 1 final assignment (60%). The grades in MyCourses (MC) are in the scale 0–100, and they are converted to a SISU grade according to the following rules:

$$\begin{aligned} 5 &\leftarrow [88, 100) \text{ MC} \\ 4 &\leftarrow [76,\ 88) \text{ MC} \\ 3 &\leftarrow [64,\ 76) \text{ MC} \\ 2 &\leftarrow [52,\ 64) \text{ MC} \\ 1 &\leftarrow [40,\ 52) \text{ MC} \\ 0 &\leftarrow [\ 0,\ 40) \text{ MC} \end{aligned} \tag{12}$$

You must compile the solutions to your homeworks/assignments in a **short report** (a PDF file) and submit them to MyCourses, before the deadline (see the schedule in the table above). We fix the deadlines to every next Sunday, so that there are two exercise sessions available to ask for help in the assignments. The final assignment is a larger problem-oriented homework which covers all the topics in the course, and you will have around 2–3 weeks to submit your final report. There is no exam.

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

[4] 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.

[5] D. E. Seborg et al. *Process dynamics and control*. John Wiley & Sons, 2016.

[6] F. Dörfler et al. "Toward a systems theory of algorithms". In: *IEEE Control Systems Letters* 8 (2024), pp. 1198–1210.