CHEM-E7225
2023

Aalto University

Advanced process control

CHEM-E7225 (was E7190), 2023

Francesco Corona (¬_¬)

Chemical and Metallurgical Engineering
School of Chemical Engineering

Overview

We study the mathematical principles and basic numerical tools for optimal control

- Understanding of optimal control
- Examples from chemical systems

- (Catchy image from the internet)

The approach is general, with application domains in many (bio)-chemical systems

Overview (cont.)

Process control and automation at Aalto University

Francesco Corona

- ⇝ Professor of process control and automation
- Once and future (on going) chem eng
- Camouflaged as computer scientist

Research and teaching about **computational** and **inferential thinking** of process systems

- ⇝ **Automatic control and machine learning**
  - **Three doctoral students**
  - **Two master's student**

Formal methods from automatic control, statistics, and optimisation, plus applications

Overview (cont.)

Optimal control studies the mathematics in the optimisation of dynamical systems

- Numerical optimisation and system theory and numerical simulation

We start by setting the basic preliminaries associated with these two fields

- ⇝ Dynamic modelling and numerical simulations, state-space form
- ⇝ Numerical optimisation with Newton-type methods

These two fields are combined to study the two flavours of optimal control

- ⇝ Discrete and continuous-time optimal control

Direct- and indirect methods, and the Hamilton-Jacobi-Bellman equation

The objective is to provide some practical introduction to optimal control

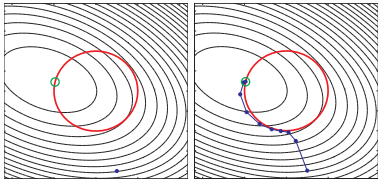
Trajectory

We study practical **optimisation** of **dynamical systems**, the basics of **dynamic optimisation**

- I Control variables and disturbances
- O Measurement variables, the data
- ⇝ The system evolves in time
- ⇝ (System/model/process)

Mathematical optimisation refers to the problem of determining the best, or optimal, solution to some problem, a definition of optimality, given a set of possible decisions

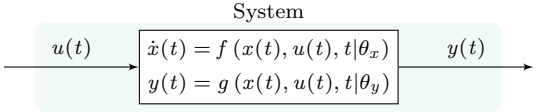
- ⇝ Decision variables
- ⇝ Objective function
- ⇝ Constraint functions



Trajectory | System dynamics

We understand dynamic systems as processes that are evolving in time and that can be represented using states that allow to determine the future behaviour of the system

I/O Controls u, outputs y
S State variables x
P Parameters $\theta_{x,y}$



The dynamics of the state vars are represented by some nonlinear function f

- Function f returns $\dot{x}(t)$ at time t, given $x(t)$, $u(t)$, and t
- Function f is parameterised, the vector θ_x

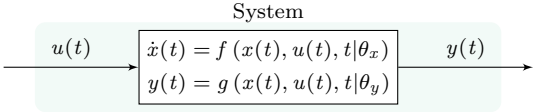
How the state vars are transformed into measurements, nonlinear function g

- Function g returns $y(t)$ at time t, given $x(t)$, $u(t)$, and t
- Function g is parameterised, the vector θ_y

Trajectory (cont.)

I/O Inputs u, outputs y

S State variables x



Functions f often from a process modelling effort: Mass/energy/momentum balances

$$\underbrace{[\text{Stuff in}] - [\text{Stuff out}] \;\pm\; [\text{Stuff generated/consumed}]}_{f(x,u,t)} = \underbrace{[\text{Stuff accumulated}]}_{\dot{x}}$$

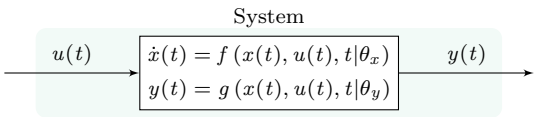
Functions g often determined by the automation system: Sensors and instruments

---

Knowledge of the initial state $x(t_0)$ and of the control trajectory $u(t)$ over some time interval such that $t \in [t_0,\, T]$ allows to determine the state trajectory $x(t)$ for $t \in [t_0,\, T]$

- The system model with certain evolution is of the deterministic kind
- Stochastic models describe evolutions that are known statistically

Trajectory | System dynamics (cont.)



One fundamental element of dynamical systems is represented by state $x(t)$ at time t

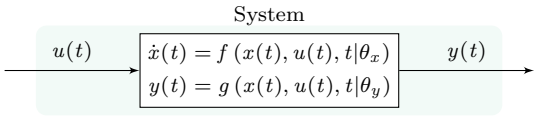
- The state space $\mathcal{X}$ is the set of all possible values of the state
- It can be continuous, like the usual $\mathcal{R}^{N_x}$ or some manifold
- It can be a discrete countable set such that $|\mathcal{X}| = N_{\mathcal{X}}$
- It can be hybrid, with continuous and discrete states

The other main element of dynamical systems is represented by controls $u(t)$ at time t

- The control/action space $\mathcal{U}$ is the set of all possible controls
- It can be continuous, like the usual $\mathcal{R}^{N_u}$ or some manifold
- It can be a discrete countable set such that $|\mathcal{U}| = N_{\mathcal{U}}$
- It can be hybrid, with continuous and discrete actions

Trajectory (cont.)

I/O Inputs u, outputs y
S State variables x



The dynamical system can be controlled by a suitable choice of inputs denoted controls

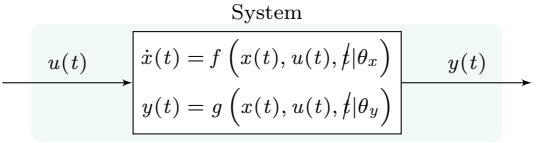
- The (sequence of) controls should be chosen optimally, in some sense
- The chosen controls must satisfy certain constraints

This course is about methods and solutions to determine the optimal control inputs

Trajectory (cont.)

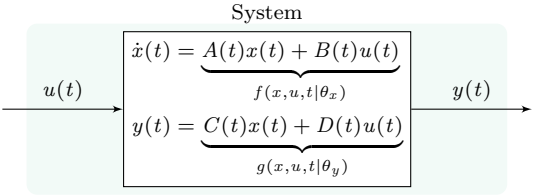
In general, function f and g may change in time (or, f and g will change with time t)

- Typical of a process operated under varying conditions or subjected to ageing
- We will not discuss such processes explicitly, focus on time-invariant systems



Sometimes functions f and g may be approximated to be linear functions in x and u

- Typical of a process model linearised around some steady-state (x^*, u^*)
- Not discussed explicitly neither, focus on nonlinear systems



Trajectory | System dynamics (cont.)

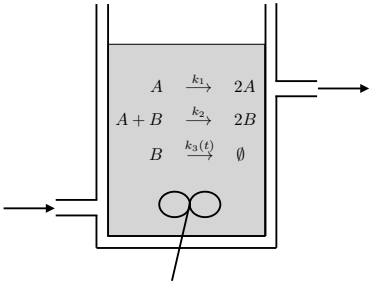
Chemical kinetics are concerned with understanding the evolution of reaction systems

- The system is specified by a set of coupled chemical reactions

The chemical kinetics induce an ordinary differential equation, a dynamical system

Trajectory | System dynamics (cont.)

Example



The Lotka-Volterra system is usually used to capture interactions between competing chemical species

$$\begin{aligned} A &\xrightarrow{k_1} 2A \\ A + B &\xrightarrow{k_2} 2B \\ B &\xrightarrow{k_3(t)} \emptyset \end{aligned}$$

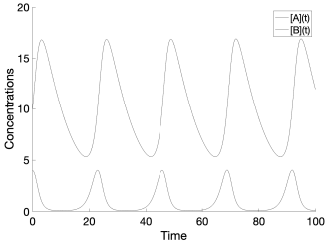
Assuming to be able directly manipulate the rate at which component B is removed,

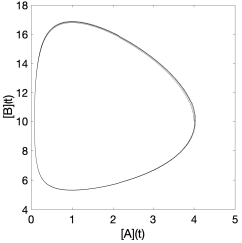
$$\begin{bmatrix} \dfrac{d[A](t)}{dt} \\ \dfrac{d[B](t)}{dt} \end{bmatrix} = \begin{bmatrix} k_1[A](t) - k_2[A](t)[B](t) \\ k_2[A](t)[B](t) - k_3(t)[B](t) \end{bmatrix}$$

Trajectory | System dynamics (cont.)

$$\underbrace{\begin{bmatrix} \dfrac{d[A](t)}{dt} \\ \dfrac{d[B](t)}{dt} \end{bmatrix}}_{\dot{x}(t)} = \underbrace{\begin{bmatrix} k_1[A](t) - k_2[A](t)[B](t) \\ k_2[A](t)[B](t) - k_3(t)[B](t) \end{bmatrix}}_{f(x(t),u(t)|\theta_x)}$$

- State variables $x(t) = (x_1(t), x_2(t)) = ([A](t), [B](t))$
- Control variables $u(t) = u_1(t) = k_3(t)$
- Parameters $\theta_x = (k_1, k_2)$

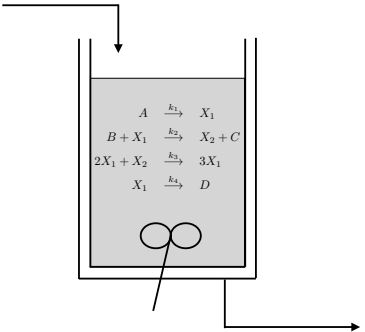




The evolution, for an initial condition $x(0) = x_0$ and sequence of inputs $u(0 \rightsquigarrow T)$ □

Trajectory | System dynamics (cont.)

Example



The Brusellator is a theoretical model for certain auto-catalytic reactions

$$
\begin{aligned}
A &\xrightarrow{k_1} X_1 \\
B + X_1 &\xrightarrow{k_2} X_2 + C \\
2X_1 + X_2 &\xrightarrow{k_3} 3X_1 \\
X_1 &\xrightarrow{k_4} D
\end{aligned}
$$

The reaction components (X_1, X_2) are a pair of intermediate state variables

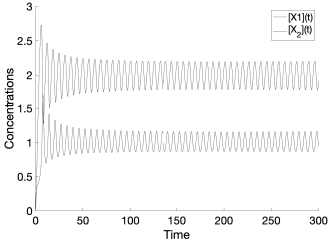
Assuming that the concentration species A and B (and C and D) can be manipulated

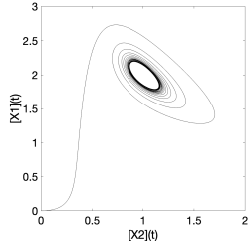
$$
\begin{bmatrix} \dfrac{d[X_1](t)}{dt} \\ \dfrac{d[X_2](t)}{dt} \end{bmatrix} = \begin{bmatrix} a(t) - b(t)[X_1](t) + [X_1]^2(t)[X_2](t) - [X_1](t) \\ b(t)[X_1](t) - [X_1]^2(t)[X_2](t) \end{bmatrix}
$$

Trajectory | System dynamics (cont.)

$$\underbrace{\begin{bmatrix} \dfrac{d[X_1](t)}{dt} \\ \dfrac{d[X_2](t)}{dt} \end{bmatrix}}_{\dot{x}(t)} = \underbrace{\begin{bmatrix} a(t) - b(t)[X_1](t) + [X_1]^2(t)[X_2](t) - [X_1](t) \\ b(t)[X_1](t) - [X_1]^2(t)[X_2](t) \end{bmatrix}}_{f(x(t),u(t)|\theta_x)}$$

- State variables $x(t) = (x_1(t), x_2(t)) = ([X_1](t), [X_2](t))$
- Control variables $u(t) = (u_1(t), u_2(t)) = (a(t), b(t))$
- Parameters $\theta_x = (k_1, k_2, k_3, k_4)$





The evolution, for an initial condition $x(0) = x_0$ and sequence of inputs $u(0 \rightsquigarrow T)$ □

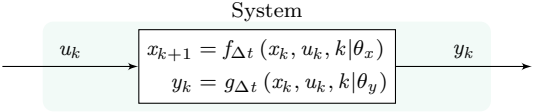
Trajectory | System dynamics (cont.)

We mainly focus on deterministic systems with continuous state and control spaces

- We will make a short digression to discuss discrete state- and action-spaces

More importantly, we are interested in solutions to be implemented on a computer

- We consider digitally controlled sampled-data system, in discrete-time



The evolution of discrete-time systems takes values on a predefined time grid

- We will often replace the continuous time variable $t \in \mathcal{R}_{\geq 0}$
- Instead, we will use the discrete index variable $k \in \mathcal{N}_0$

Trajectory | System dynamics (cont.)

Overview of the main (useful for us) classes of dynamical system models

- Continuous- and discrete-control
- Continuous- and discrete-state
- Continuous- and discrete-time

Important properties: Stability, controllability (and observability)

Existence and uniqueness of the solution to initial value problems

- Picard-Lindelöf and local existence and uniqueness

Numerical simulation (or, integration) of initial value problems

- Zero-order hold of the control and the solution map
- Explicit integration (Euler and Runge-Kutta)

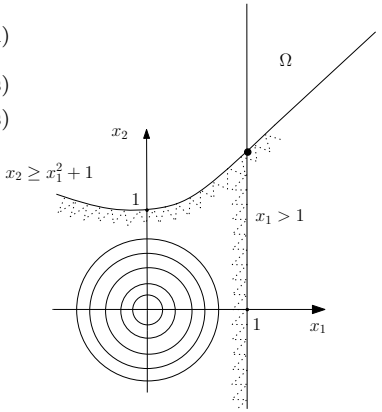
Coding/simulating interesting dynamical system

⇝ Assignment I

Trajectory | Optimisation

Mathematical optimisation refers to the task of finding the best solution x^* to a problem, an optimality definition for some function $f(x)$, in a set of feasible decisions x

$$\begin{aligned} \min_{x \in \mathcal{R}^N} \quad & f(x) && \text{(Objective function)} \\ \text{subject to} \quad & g(x) = 0 && \text{(Equality constraints)} \\ & h(x) \geq 0 && \text{(Inequality constraints)} \end{aligned}$$



We mainly consider finite-dimensional optimisation tasks with smooth functions

- $f : \mathcal{R}^N \to \mathcal{R}$, with $f \in \mathcal{C}^2\left(\mathcal{R}^N\right)$
- $g : \mathcal{R}^N \to \mathcal{R}^{N_g}$, with $g \in \mathcal{C}^2\left(\mathcal{R}^N\right)$
- $h : \mathcal{R}^N \to \mathcal{R}^{N_h}$, with $h \in \mathcal{C}^2\left(\mathcal{R}^N\right)$

Continuity of the search domain and smoothness of the objective and constraint functions allow the use derivative-based approaches, applications of the Newton's method

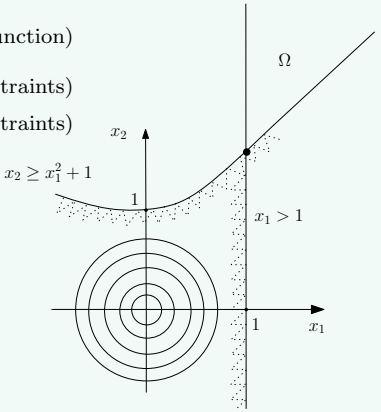
Trajectory | Optimisation (cont.)

Example

$$\min_{x \in \mathcal{R}^N} \quad x_1^2 + x_2^2 \qquad \text{(Objective function)}$$

$$\text{subject to} \quad x_1 - 1 = 0 \qquad \text{(Equality constraints)}$$

$$x_2 - 1 - x_1^2 \geq 0 \qquad \text{(Inequality constraints)}$$



- $f : \mathcal{R}^2 \to \mathcal{R}$, with $f \in \mathcal{C}^2\left(\mathcal{R}^2\right)$
- $g : \mathcal{R}^2 \to \mathcal{R}$, with $g \in \mathcal{C}^2\left(\mathcal{R}^2\right)$
- $h : \mathcal{R}^2 \to \mathcal{R}$, with $h \in \mathcal{C}^2\left(\mathcal{R}^2\right)$

The set of feasible decisions

$$\Omega = \{x \in \mathcal{R}^2 | h(x) \geq 0, g(x) = 0\}$$

The minimiser x^* (here, point •)

Trajectory | Optimisation (cont.)

Newton- and Newton-type methods are numerical techniques for root-finding

Consider some vector-valued function $F : \mathcal{R}^N \rightarrow \mathcal{R}^N$

$$F(x) = \begin{bmatrix} f_1(x_1, x_2, \ldots, x_N) \\ f_2(x_1, x_2, \ldots, x_N) \\ \vdots \\ f_N(x_1, x_2, \ldots, x_N) \end{bmatrix}$$

Newton methods are used to find the zeros of F

- Points $x^* \in \mathcal{R}^N$ where $F(x) = 0$

$$F(x^*) = \begin{bmatrix} f_1(x_1^*, x_2^*, \ldots, x_N^*) \\ f_2(x_1^*, x_2^*, \ldots, x_N^*) \\ \vdots \\ f_N(x_1^*, x_2^*, \ldots, x_N^*) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Many optimisation problems can be formulated as root-finding problems

$\rightsquigarrow$ Newton's methods are the basis for their numerical solution

Trajectory | Optimisation (cont.)

$$\begin{aligned} \min_{x \in \mathcal{R}^N} \quad & f(x) && \text{(Objective function)} \\ \text{subject to} \quad & g(x) = 0 && \text{(Equality constraints)} \\ & h(x) \geq 0 && \text{(Inequality constraints)} \end{aligned}$$

The constraint set $\Omega \subseteq \mathcal{R}^N$ is specified in terms of equality and inequality constraints

We can take into account this structure to obtain a collection of optimality conditions

- These conditions involve a set of auxiliary variables, $\lambda \in \mathcal{R}^{N_h}$ and $\mu \in \mathcal{R}^{N_g}$
- The auxiliary variables are denoted as Lagrange multipliers

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \sum_{n_g=1}^{N_g} \mu_{n_g} g_{n_g}(x) - \sum_{n_h=1}^{N_h} \lambda_{n_h} h_{n_h}(x)$$

The theory of Lagrange multipliers facilitates the characterisation of optimal solutions

- The Lagrange function $\mathcal{L}$ and the primal and dual formulation
- The Karhush-Kuhn-Tucker (KKT) optimality conditions

Trajectory | Optimisation (cont.)

Overview of the main (useful for us) classes of mathematical programming problems

- Nonlinear programming
- Quadratic programs
- (Linear programs)

The critical boundary in optimisation is between convex and non-convex programs

Newton- and Newton-type root-finding methods and unconstrained optimisation

- Generalities, optimality conditions, and convergence

Existence and optimality conditions of the solution to a nonlinear program

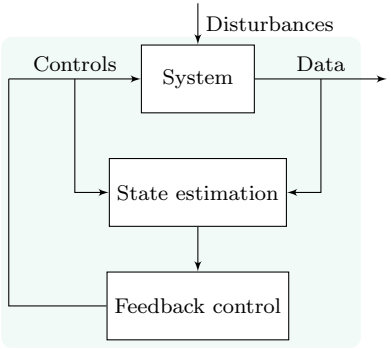
- Equality constrained optimisation problems
- Inequality constrained optimisation

Numerical algorithms and coding

⇝ Assignment II

Trajectory | Dynamic optimisation

Optimal state-feedback control combines notions of optimisation and system dynamics

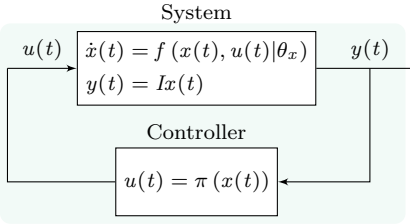


A general structure, each block can be treated using different technologies

- First-principles (physics)
- Empirical (data-derived)

It scales (it can be solved) reasonably well with the process size (complexity)

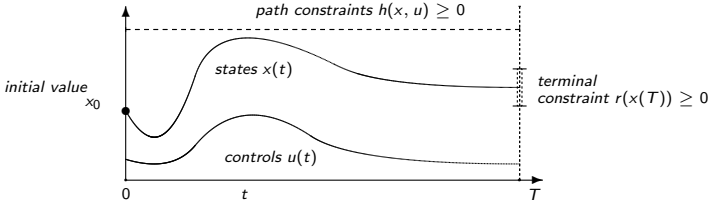
Trajectory | Dynamic optimisation (cont.)



- $\rightsquigarrow$ A (known) dynamical system
- $\rightsquigarrow$ A formal control objective
- $\rightsquigarrow$ A set of constraints

Find the best control function $u(0 \rightsquigarrow T)$ that optimally achieves the objective, while satisfying the constraints

---



Trajectory | Dynamic optimisation (cont.)

$$\min_{\substack{x(0\rightsquigarrow T)\\ u(0\rightsquigarrow T)}} \underbrace{\underbrace{E\left(x(T)\right)}_{\text{Mayer term}} + \underbrace{\int_0^T L\left(x(t),u(t)\right)\,dt}_{\text{Lagrange term}}}_{\text{Bolza term}}$$

$$\begin{aligned}
\text{subject to}\quad & x(0) - \overline{x}_0 = 0, && t = 0 && \text{(fixed initial state)}\\
& \dot{x}(t) - f\left(x(t),u(t)\right) = 0, && t \in [0,\,T] && \text{(dynamics)}\\
& h\left(x(t),u(t)\right) \geq 0, && t \in [0,\,T] && \text{(path constraints)}\\
& r\left(x(T)\right) \geq 0, && t = T && \text{(terminal constraints)}
\end{aligned}$$

Continuous-time optimal control problem, it is an ∞-dimensional optimisation

- The decision variables are functions of time, $u(0 \rightsquigarrow T)$ and $x(0 \rightsquigarrow T)$

Bolza objective functions are the sum of two terms, a Lagrange and a Mayer term

- The integral cost $L\left(x(t),u(t)\right)$ over $[0,\,T]$ and a terminal cost $E\left(x(T)\right)$

Trajectory | Dynamic optimisation (cont.)

$$\min_{\substack{x(0\rightsquigarrow T)\\ u(0\rightsquigarrow T)}} \quad \underbrace{\underbrace{E\left(x(T)\right)}_{\text{Mayer term}} + \underbrace{\int_0^T L\left(x(t),u(t)\right)\,dt}_{\text{Lagrange term}}}_{\text{Bolza term}}$$

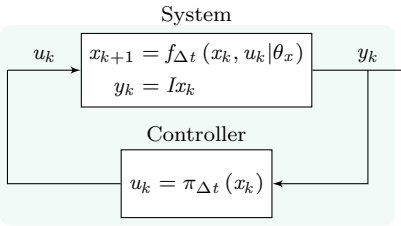
$$\begin{aligned}
\text{subject to} \quad & x(0) - \overline{x}_0 = 0, && t = 0 && \text{(fixed initial state)}\\
& \dot{x}(t) - f\left(x(t),u(t)\right) = 0, && t \in [0,\,T] && \text{(dynamics)}\\
& h\left(x(t),u(t)\right) \geq 0, && t \in [0,\,T] && \text{(path constraints)}\\
& r\left(x(T)\right) \geq 0, && t = T && \text{(terminal constraints)}
\end{aligned}$$

Three main classes of approaches to solve continuous-time optimal control problems

- Principle of optimality (the Hamilton-Jacobi-Bellman equation)
- Calculus of variations (the Pontryagin maximum principle)
- **Discretise-then-optimise**, then nonlinear programming

Trajectory | Dynamic optimisation (cont.)

Discrete-time optimal control



- A (known) dynamical system
- A formal control objective
- A set of constraints

Find the best control sequence $u_0, u_1, \ldots, u_{K-1}$ that optimally achieves the objective

$$\min_{\substack{x_0, x_1, \ldots, x_K \\ u_0, u_1, \ldots, u_{K-1}}} \quad E(x_K) + \sum_{k=0}^{K-1} L(x_k, u_k)$$

subject to

$$\begin{aligned}
& x_0 - \overline{x}_0 = 0, && k = 0 && \text{(fixed initial state)} \\
& x_{k+1} - f_{\Delta t}(x_k, u_k) = 0, && k = 0, 1, \ldots, K-1 && \text{(dynamics)} \\
& h(x_k, u_k) \geq 0, && k = 0, 1, \ldots, K-1 && \text{(path constraints)} \\
& r(x_K) \geq 0, && k = K && \text{(terminal constraints)}
\end{aligned}$$

Trajectory | Dynamic optimisation (cont.)

The formulation of the discrete-time optimal control is a general nonlinear program

$$\min_{\substack{x_0,x_1,\ldots,x_K \\ u_0,u_1,\ldots,u_{K-1}}} \quad \underbrace{E\left(x_K\right)}_{\text{Terminal cost}} + \sum_{k=0}^{K-1} \underbrace{L\left(x_k,u_k\right)}_{\text{Stage cost}}$$

$$\begin{aligned}
\text{subject to} \quad & x_0 - \overline{x}_0 = 0, && k = 0 && \text{(fixed initial state)} \\
& x_{k+1} - f_{\Delta t}\left(x_k,u_k\right) = 0, && k = 0,\ldots,K-1 && \text{(dynamics)} \\
& h\left(x_k,u_k\right) \geq 0, && k = 0,\ldots,K-1 && \text{(path constraints)} \\
& r\left(x_K\right) \geq 0, && k = K && \text{(terminal constraints)}
\end{aligned}$$

The number of decision variables is typically very large, $(K \times \mathcal{R}^{N_x}) \times ((K-1) \times \mathcal{R}^{N_u})$

- Optimisation can be approached with solvers for generic nonlinear programs
- The problem has however a sparsity structure that can be exploited

This discrete problem is solved by using optimisation and integration simultaneously

Trajectory | Dynamic optimisation (cont.)

In the simultanous approach, both control and state variables are decision variables

It is possible to eliminate nearly all the state variables, by using the dynamics

$$\begin{aligned}
\widetilde{x}_0(x_0) &= x_0 \\
\widetilde{x}_1(x_0, u_0) &= f_{\Delta t}\left(\widetilde{x}_0(x_0), u_0\right) \\
\widetilde{x}_2(x_0, u_0, u_1) &= f_{\Delta t}\left(\widetilde{x}_1(x_0, u_0), u_1\right) \\
\cdots &= \cdots \\
\widetilde{x}_{k+1}(x_0, u_0, u_1, \cdots, u_{k-1}, u_k) &= f_{\Delta t}\left(\widetilde{x}_k(x_0, u_0, \ldots, u_{k-1}), u_k\right) \\
\cdots &= \cdots
\end{aligned}$$

We obtain the reduced formulation,

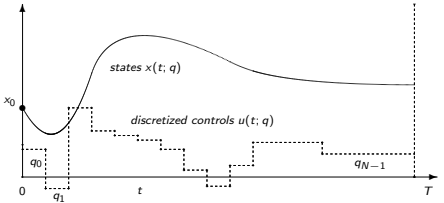
$$\begin{aligned}
\min_{\substack{x_0 \\ u_0, u_1, \ldots, u_{K-1}}} \quad & E\left(\overline{x}_K\left(x_0, u_0, u_1, \ldots, u_{K-1}\right)\right) + \sum_{k=0}^{K-1} L\left(\overline{x}_k\left(x_0, u_0, u_1, \ldots, u_{K-1}\right), u_k\right) \\
\text{subject to} \quad & x_0 - \overline{x}_0 = 0 \\
& h\left(x_k, u_k\right) \geq 0 \\
& r\left(x_K\right) \leq 0
\end{aligned}$$

Many less decision variables $(N_x + (K-1)N_u)$, but the sparsity structure is disrupted

Trajectory | Dynamic optimisation (cont.)

Direct optimal control methods transcribe the original infinite-dimensional optimisation problem into a finite-dimensional one to be solved by nonlinear programming

- Define a finite-dimensional parameterisation of the control trajectory

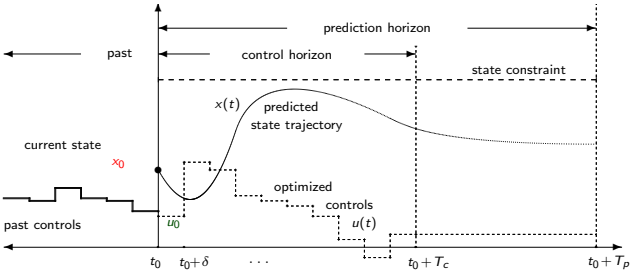


The idea is to discretise the controls $u(t)$ on a fixed grid $0 = t_0 < t_1 < \cdots < T_K = T$

- Controls are parameterised by polynomials (often, piecewise constant functions)

⇝ Assignment III

Trajectory | Model predictive control



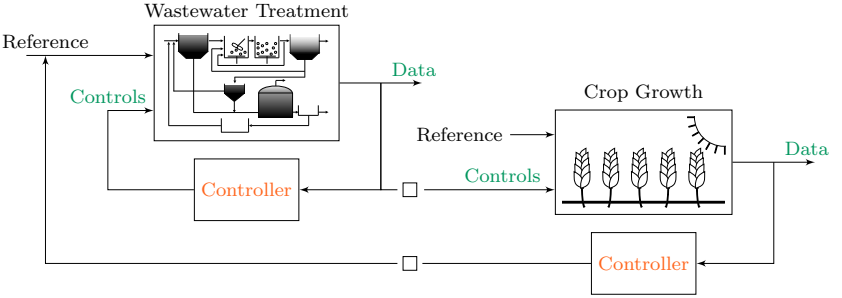
$$\min_{u_k,\ldots,u_{k+N-1}} \quad \sum_{n=k}^{k+N-1} L(x_n, u_n) + L_f(x_{k+N})$$

$$\text{s.t.} \quad x_{n+1} = f_{\Delta t}(x_n, u_n, w_n|\theta_x),$$
$$\forall n \in [k, k+N]$$

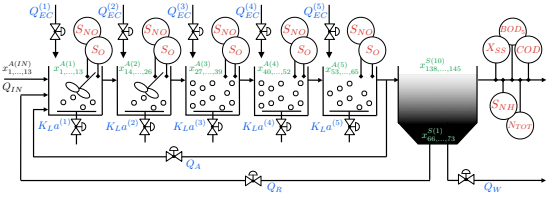
$$x_n \in \mathcal{X}, \quad u_n \in \mathcal{U},$$
$$\Phi(x_k, x_{k+N}) = 0,$$

At each time step, solve the finite-horizon optimal control and then apply only the first optimal control action

Trajectory | Model predictive control (cont.)



- State vars
- Controls
- Disturbances
- Measurements



Neto et al. ECC (2020), IFAC (2020), ADCHEM (2021), JPC (2022), DYCOPS (2022)

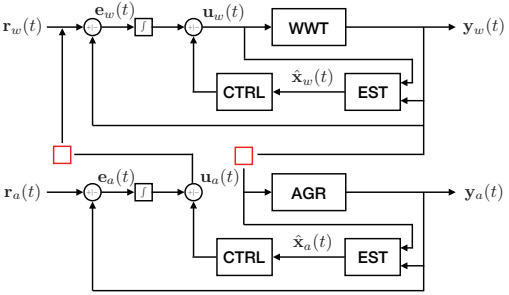
Trajectory | Model predictive control (cont.)

⇝ **Hamilton-Jacobi-Bellman**

- **On AGR**

⇝ **Nonlinear programming**

- **On WWT**



A guest seminar on model predictive control of wastewater treatment plants for reuse

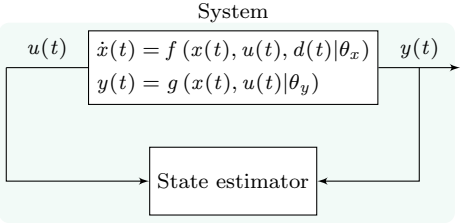
- Online optimal control for tracking, disturbance rejection, and zero offset control
- Date yet to be set (last or last-but-one week)

Trajectory | Moving horizon estimation

The variables that are technologically/economically measurable are a small subset

- The state variables of the system are often not measured

- Measurements are corrupted by noise
- The dynamics are not fully known
- Disturbances drive the evolution



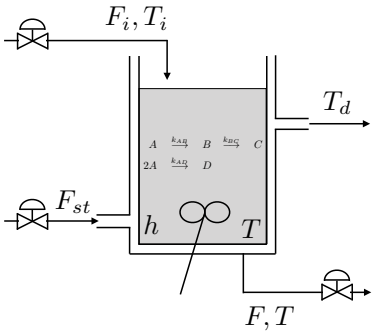
State estimation for use in control in the presence of sensor and model uncertainty

- Classic stochastic approaches, the Kalman and the extended Kalman filter

We discuss a deterministic counterpart for state estimation, only based on optimisation

$$\left(\|x(0) - \overline{x}_0\|^2_{P^{-1}(0)}\right) + \left(\sum_{k=0}^{K-1} \|x_{k_+1} - f_{\Delta t}\left(x_k, u_k, d_k|\theta_x\right)\|^2_{Q^{-1}}\right) + \left(\sum_{k=0}^{K} \|y_k - g\left(x_k, u_k, d_k|\theta_y\right)\|^2_{R^{-1}}\right), \quad \text{s.t. } x_k \in \mathcal{X}, u_k \in \mathcal{U}$$

Trajectory | Moving horizon estimation (cont.)



$$A \xrightarrow{k_{AB}} B \xrightarrow{k_{BC}} C$$
$$2A \xrightarrow{k_{AD}} D$$

A non-isothermal CSTR (Van de Vusse)

- Nonlinear and non-minimum phase

$$\frac{d\rho_A}{dt} = q\left(\rho_{\text{in}}^{(A)} - \rho_A\right) - \left(k_{AB}(T)\rho_A + k_{AD}(T)\rho_A^2\right)$$
$$\frac{d\rho_B}{dt} = -q\rho_B + k_{AB}(T)\rho_A - k_{BC}(T)\rho_B$$
$$\frac{dT}{dt} = q\,(T_{\text{in}} - T) + \frac{k_W A_r}{\varrho C_p V_r}(T_d - T) - \frac{1}{\varrho C_p}\left(k_{AB}(T)\rho_A \Delta H_{AB} + k_{BC}\rho_B \Delta H_{BC} + k_{AB}(T)\rho_A^2 \Delta H_{AC}\right)$$
$$\frac{dT_d}{dt} = \frac{1}{m_K C_{pK}} Q + \frac{k_W A_r}{m_K C_{pK}}(T - T_d)$$

Another guest seminar, on the moving horizon estimator (Date to be set)

Overview

CHEM-E7225 is a set of **lectures (approx. 36h)** and **exercises (approx. 12h)**

W02	**Dynamical systems and simulation**		
	L		Intro
	L		Dynamical systems and simulation
	L		Dynamical systems and simulation
	E		Exercise (set-up coding environment)
W03	**Optimisation**		
	E+L		Root-finding with Newton's methods
	L		Nonlinear programming
	L		Nonlinear programming
	E+L		Exercise (Simulation)
W04	**Optimisation**		
	E+L		Nonlinear programming
	L		Nonlinear programming
	L		Discrete-time optimal control
	E		Exercise (Root-finding/Optimisation)
W05	**Dynamic optimisation**		
	E+L		Discrete-time optimal control
	L		Dynamic programming
	L		Dynamic programming
	E		Exercise (Optimisation)
W06	**Topic**		
	E+L		Linear-quadratic and infinite-horizon problems
	L		Seminar (Model-predictive control)
	L		The Hamilton-Jacobi-Bellman equation
	E		Exercise (Discrete-time optimal control)
W07	**Topic**		
	E+L		Continuous-time optimal control
	L		Seminar (Moving-horizon estimation)
	L		Continuous-time optimal control
	E		Exercise (Continuous-time optimal control)

Lectures/exercise schedule will be modified to accommodate the class needs

Overview (cont.)

To pass E7225 you must return all the exercises (80%) and participate (20%)

Exercises (80%)

- One (1) written report with your solutions
- Include your results and your code
- Include high-quality diagrams
- Discuss your solution/code

⇝ By **FEB 28, 23:59:59**

Upload a (1) single (1) file, only use PDFs[1]

Participation (20%)

- Engage with the course activities
- Comment on the lecture notes
- Find and report typos/bugs

We encourage you to collaborate in figuring out answers and help others solve the problems, yet we ask you to submit your work individually and to explicitly acknowledge those with whom you collaborated. We are assuming that you take the responsibility to make sure you personally understand the solution to work arising from collaboration

[1] If you have multiple files, merge them. If you use MSWord or else, save as PDF.